Universität Bremen

University of Bremen
Fachbereich 3 – Robotics Group

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH
Robotics Innovation Center

German Research Center for Artificial Intelligence
Robotics Innovation Center

An Intelligent Architecture for Legged Robot Terrain Classification Using Proprioceptive and Exteroceptive Data

Doctoral Dissertation in partial fulfillment of the requirements for the degree doctor of Philosophy in engineering "**Dr. Ing.**"

by
Mohammed Nour Abdel Gwad Ahmed

MSc in Computer and Control Engineering
Faculty of Engineering, Zagazig University, Egypt

Advisor
Prof. Dr. Frank Kirchner

Head of Robotics Group, University of Bremen
Director of DFKI Bremen–Robotics Innovation Center

Bremen, 2015

Doctoral Dissertation submitted to University of Bremen
in partial fulfillment of the requirements for the degree:
Doctor of Philosophy
in engineering "**Dr. Ing.**"

Adviser
Prof. Dr. Frank Kirchner
Head of Robotics Group, University of Bremen
Director of DFKI Bremen–Robotics Innovation Center

Additional Doctoral Committee Members
Prof. Dr.-Ing. Udo Frese
University of Bremen, Fachbereich 3
head of multi-sensorial interactive systems Group

Prof. Dr. Michael Beetz
University of Bremen, Faculty for Informatics
head of the Institute for Artificial Intelligence

Prof. Dr. Matthew Hölzel
University of Bremen, Faculty for Informatics
head of the Parallel Computing for Embedded Sensor Systems Research Group

Abraham Temesgen Tibebu
University of Bremen, Fachbereich 3
Researcher at the Robotics Innovation Center

Felix Lüdeke
University of Bremen, Fachbereich 1
Student in sixth semester at Faculty of Physics and Electrical Engineering

Date of Defense
24.06.2015

Revision SVN.198.50.20150725

Universität Bremen

Dissertation

An Intelligent Architecture for Legged Robot Terrain Classification Using Proprioceptive and Exteroceptive Data

zur Erlangung des Grades eines
Doktors der Ingenieurwissenschaften
Dr. Ing.

von

Mohammed Nour Abdel Gwad Ahmed

MSc in Computer and Control Engineering
Faculty of Engineering, Zagazig University, Egypt

Vorgelegt im Fachbereich 3
der Universität Bremen
im Juni 2015

Vom Promotionsausschuss der
Universität Bremen
zur Erlangung des akademischen Grades
Doktor-Ingenieur (Dr.-Ing.)
genehmigte Dissertation

Datum des Promotionskolloquiums
24. Juni 2015

Gutachter
Prof. Dr. Frank Kirchner
Leiter der Arbeitsgruppe Robotik, FB3, Universität Bremen
Direktor des DFKI Bremen–Robotics Innovation Center

Prof. Dr.-Ing. Udo Frese
Leiter der Arbeitsgruppe Multisensorische Interaktive Systeme,
Fachbereich 03 - Mathematik und Informatik, Universität Bremen

Weitere Mitglieder des Prüfungsausschusses
Prof. Dr. Michael Beetz
Prof. Dr. Matthew Hölzel
Abraham Temesgen Tibebu
Felix Lüdeke

to

Yasmin, **Nour**, **Adham**, and **Lena**.

The best in my life and after ...

Abstract

In this thesis, we introduce a novel architecture called "Intelligent Architecture for Legged Robot Terrain Classification Using Proprioceptive and Exteroceptive Data (*i*ArteC)". The proposed architecture integrates different terrain characterization and classification with other robotic system components.

Within *i*ArteC , we consider the problem of having a legged robot autonomously learn to identify different terrains. Robust terrain identification can be used to enhance the capabilities of legged robot systems, both in terms of locomotion and navigation. For example, a robot that has learned to differentiate sand from gravel can autonomously modify (or even select a different) path in favor of traversing over a *better* terrain. The same knowledge of the terrain type can also be used to guide a robot in order to avoid specific terrains. To tackle this problem, we developed four approaches for terrain characterization, classification, path planning, and control for a mobile legged robot.

We developed a particle system inspired approach to estimate the robot foot–ground contact interaction forces. The approach is derived from the well known Bekker's theory to estimate the contact forces based on its point contact model concepts. It is realistically model real-time 3-dimensional contact behaviors between rigid body objects and the soil. For a real-time capable implementation of this approach, its reformulated to use a lookup table generated from simple contact experiments of the robot foot with the terrain.

Also, we introduced a short-range terrain classifier using the robot embodied data. The classifier is based on a supervised machine learning approach to optimize the classifier parameters and terrain it using proprioceptive sensor measurements. The learning framework preprocesses sensor data through channel reduction and filtering such that the classifier is trained on the feature vectors that are closely associated with terrain class.

For the long–range terrain type prediction using the robot exteroceptive data, we present an online visual terrain classification system. It uses only a monocular camera with a feature-based terrain classification algorithm which is robust to changes in illumination and view points. For this algorithm, we extract local features of terrains using Speed Up

Robust Feature (SURF). We encode the features using the Bag of Words (BoW) technique, and then classify the words using Support Vector Machines (SVMs).

In addition, we described a terrain dependent navigation and path planning approach that is based on E* planer and employs a proposed metric that specifies the navigation costs associated terrain types. This generated path naturally avoids obstacles and favors terrains with lower values of the metric. At the low level, a proportional input-scaling controller is designed and implemented to autonomously steer the robot to follow the desired path in a stable manner.

*i*ARTEC performance was tested and validated experimentally using several different sensing modalities (proprioceptive and exteroceptive) and on the six legged robotic platform CREX. The results show that the proposed architecture integrating the aforementioned approaches with the robotic system allowed the robot to learn both robot-terrain interaction and remote terrain perception models, as well as the relations linking those models. This learning mechanism is performed according to the robot own embodied data. Based on the knowledge available, the approach makes use of the detected remote terrain classes to predict the most probable navigation behavior. With the assigned metric, the performance of the robot on a given terrain is predicted. This allows the navigation of the robot to be influenced by the learned models.

Finally, we believe that *i*ARTEC and the methods proposed in this thesis can likely also be implemented on other robot types (such as wheeled robots), although we did not test this option in our work.

Zusammenfassung

In dieser Arbeit wird eine neue “Intelligente Architektur zur Geländeklassifikation für laufende Roboter unter Nutzung propriozeptiver und exterozeptiver Informationen (*i*ARTEC)” vorgestellt. Die vorgeschlagene Architektur integriert verschiedene Charakterisierungen und Klassifizierungen des Geländes in ein robotisches System.

In *i*ARTEC betrachten wir das Problem eines laufenden Roboters, die Unterscheidung verschiedener Geländebeschaffenheiten selbstständig zu erlernen. Eine robuste Identifikation des Geländes kann genutzt werden, um die Bewegungs- und Navigationsfähigkeiten des Laufroboters zu verbessern. So kann zum Beispiel ein Roboter, der gelernt hat Sand und Kies zu unterscheiden, autonom seinen Weg ändern (bzw. neu wählen), um sich auf *besserem* Untergrund bewegen zu können. Das selbe Wissen um die Geländebeschaffenheit kann genutz werden, um bestimmte Untergründe vollständig zu meiden.

Um dieses Problem anzugehen haben wir vier Ansätze zur Geländecharakterisierung, Klassifikation, Pfadplanung und Steuerung für einen Laufroboter entwickelt. Im Kontext dieser Arbeit zielt die Geländeklassifikation darauf, jedem Untergrund eine von mehreren vordefinierten Geländearten wie Kies, Sand oder Steinboden zuzuordnen. Im Gegensatz dazu zielt Geländecharakterisierung darauf, Schlüsselparameter des Untergrunds zu bestimmen, welche die dynamische Interaktion mit den Füßen des Roboters beeinflussen. Diese zwei Kategorien werden zusammen als *Identifikation* bezeichnet. Die vorgestellten Ansätze sind Methoden zur Schätzung der Interaktionskräfte zwischen Roboterfuß und Untergrund, ein Nahbereichs-Geländeklassifikator für die lokale Umgebung, eine visuelle Klassifikation entfernter Gebiete sowie eine Pfadplanung und -verfolgung.

In *i*ARTEC wurde ein Ansatz auf Basis eines Partikel-Systems zur Schätzung der Kräfte zwischen Roboterfuß und Untergrund entwickelt. Dieser Ansatz ist abgeleitet von der bekannten Bekker-Theorie zur Schätzung von Krafteinwirkungen basierend auf einem Punktkontaktmodell. Diese modelliert realistisch und in Echtzeit das dreidimensionale Kontakverhalten zwischen Festkörpern und Erdboden. Für eine echtzeitfähige Implementation dises Ansatzes wurde es so angepasst, dass eine Tabelle mit Daten aus einfachen Experimenten mit einem realen Roboterfuß und verschiedenen Böden verwendet werden kann.

Zusätzlich stellen wir einen Geländeklassifikator für den Nahbereich vor, welcher auf Basis systeminterner Daten (embodied data) arebeitet. Der Klassifikator basiert auf überwachtem Lernen zur Optimierung der Klassifikatorparameter und wird mit propriozeptiven Daten trainiert. Die Sensordaten werden durch Kanalreduktion und Filterung vorverarbeitet, so dass der Klassifikator auf diesen Merkmalsvektoren trainiert wird, die eng mit den Geländeklassen verknüpft sind.

Für die Geländebestimmung im Fernbereich auf Basis exterozeptiver Daten stellen wir ein visuelles Online-Klassifikationssystem vor. Dieses verwendet eine Kamera sowie einen merkmalbasierten Klassifikationsalgorithmus, der robust ist gegenüber Änderungen der Beleuchtung und des Blickwinkels. Dafür werden lokale Merkmale des Geländes mittels SURF (Speed Up Robust Feature) extrahiert. Die Merkmale werden anschließend mit Hilfe der Bag-of-Words (BoW) Technik kodiert und durch eine Support Vector Machine (SVM) klassifiziert.

Außerdem beschreiben wir eine geländeabhängige Navigation und Pfadplanung, die auf einem E^* -Planer basiert und ein Maß berechnet, welches die Kosten für die Überquerung eines bestimmten Geländetyps quantifiziert. Der so generierte Pfad vermeidet Hindernisse und bevorzugt Regionen mit niedrigeren Werten für das beschriebene Maß. Auf unterer Ebene wird ein "proportional input-scaling controllerëntworfen und implementiert, mit dem der Roboter autonom einem gegebenen Pfad folgen kann.

Die Performanz des Systems wurde experimentell unter Nutzung diverser Sensoren (propriozeptiv und exterozeptiv) auf dem 6-beinigen Laufroboter CREX getestet.

Die Ergebnisse zeigen dass die vorgeschlagene Architektur es dem Roboter ermöglicht, sowohl die Interaktion mit dem Gelände, die entfernte Geländewahrnehmung sowie die Relation dieser beiden Modelle zu erlernen. Dieser Lernmechanismus wird unter Nutzung der systeminternen Informationen ausgeführt. Basierend auf dem vorhandenen Wissen wird das wahrscheinlichste Verhalten auf entferntem Gelände vorhergesagt. Mit dem zugeordneten Maß wird die Performanz des Roboters auf diesem Gelände geschätzt. Das erlaubt es, die Navigation des Roboters durch die gelernten Modelle beeinflussen zu lassen.

Abschließend sei gesagt, dass *i*ARTEC sowie die vorgestellten Methoden mit hoher Wahrscheinlichkeit auch auf anderen Robotern (z.B. radgetriebenen Systeme) eingesetzt werden könnten, auch wenn diese Option im Rahmen unserer Arbeit nicht evaluiert wurde.

Acknowledgements

This dissertation consumed huge amount of work, research and dedication. Still, accomplishment would not have been possible if I did not have a support of many individuals. Therefore I would like to extend my sincere gratitude to all of them.

First of all, I would like to sincerely thank my advisor and institute director Prof. Dr. Frank Kirchner. He gave me the opportunity to work at a great institute with great people in great projects.

I am also grateful to my friends and colleagues, Yong-Ho Yoo, Malte Langosz, and Lorenz Quack. For getting into the basics of this thesis, I had a lot of support by Yong-Ho Yoo who raised my first interest during the initial part of this work. Lorenz me a lot with the VEROSIM simulator and experiments. Malte helped so much in simulation and optimization as he is the main developer of the fascinating robot simulator MARS.

I am indebted to Mario Michael Krell for making support vector machine easier to understand and to introduce pySPACE, the machine learning framework, to me. When needing someone for the discussion of any problem, Mario was always there and also solved a lot of pySPACE programming problems for/with me.

I express my warm thanks to Mehmed Yüksel, Florian Cordes, and Thomas Röhr for making the CREX robot available to me and the time they spend assisting me to carry out the field experiments. Without their superior knowledge and experience, the experiments would not have that like in quality of outcomes, and thus their support has been essential. I wish to express my sincere gratitude to Mehmed Yüksel for helping when I had questions as well as frustrations.

I would like to most importantly acknowledge the effort of my wife, who encouraged me to pursue higher education and support me through the difficulties associated with such a goal even when I was not sure I would make it through.

Last but not least, I would like to thank my family and friends for always encouraging me onwards. My sisters and brothers have always been very encouraging and accommodative. I can never thank them enough for their love and faith.

This work was supported by a stipend from the Egyptian Ministry of Higher Education (scholarship:2/8/47/2005/2006). I would like to thank the funders. They had no role in study design, data collection and analysis, decision to publish, or preparation of this thesis.

Contents

List of Figures

List of Tables

List of Algorithms

"There is nothing more difficult to take in hand, more perilous to conduct or more uncertain in its success than to take the lead in the introduction of a new order of things."

— Niccolo Machiavelli, (Italian writer and statesman, Florentine patriot, author of 'The Prince', 1469-1527)

1 Introduction

For the past few decades, robotic systems have played an increasing role in our society. Their first economical use was in factories, where they executed simple repetitive tasks. In these setups, the robots had no need to be able to grasp new knowledge. Everything was pre-programmed, under the assumption that their tightly controlled environment of factory floors would not change.

As technological and algorithmic improvements made over, the past decades have tremendously increased the level of intelligence that robotic systems can now display. This made it possible to deploy mobile robotic systems in increasingly challenging outdoor environments. To achieve its goals, the robot has to interact with an unknown actor, which is the environment. This effect is even more apparent in the case of applications in rough–terrain.

A substantial part of the Earth is inaccessible to any sort of wheeled mechanism. In fact, natural obstacles like large rocks, loose soil, deep ravines, and steep slopes render wheeled locomotion ineffective. The moon and other planets present similar terrain challenges. In many of these natural terrains legged robots present superior mobility in natural terrains. With each leg comprising several links connected by prismatic or rotational joints, these systems intend to mimic the biology and, consequently, present advantages over conventional vehicles that use wheels or tracks, since they can easily adapt to irregular terrains. Interested reader can check [Silva and Tenreiro Machado, 2007] for an account of the development of legged robots.

Since these robots may use discrete footholds for each foot, in contrast to wheeled vehicles, which need a continuous support surface. Therefore, legged vehicles may move over irregular terrain, by varying their legs configuration in order to adapt themselves to surface irregularities. In addition, feet may establish contact with the ground at selected points in accordance with the terrain conditions to avoid small obstacles by passing up undesirable footholds. For these reasons, legs are inherently adequate systems for locomotion over irregular ground while staying leveled and stable.

Hence we are interested in providing the robot with the capability to predict the terrain type and how its feet interact with it while it operates in a mission. The accumulated knowledge could then be used to improve the robot behavior. Thus the idea is to make the best out of an operating platform by giving it the capability to link remote data and local data.

Local *proprioceptive*[1] data expresses the robot–terrain interaction which characterizes some aspect of the robot behavior. This robot–terrain interaction model can be related to sensors such as an inertial measurement unit (IMU), actuator encoders and other proprioceptive sensors. On the other hand, remote *exteroceptive*[2] data describes the environment at a distance and is provided by sensors such as camera, LIDARs (Light Detection And Ranging), and so forth. What this data expresses is referred to as remote terrain perception.

Associating between local data (information that sensed from the terrain in direct contact with the robot feet) and remote data (information that is predicted for the far terrain by the robot exteroceptive sensors) allows anticipating the robot-terrain interaction characteristics ahead of robot position. This information can be used to influence the robot behavior by changing its path. The data association can be referred to as near-to-far [Sancho-pradel and Gao, 2010] and is a concept at the very center of our approach. In fact, this data association allows, in our case, the inference of remote data based on local data, which corresponds to generating the interaction characteristic of the terrain lying ahead of the robot.

1.1 Motivation and Research Objective

Most research on off–road mobile robot sensing focuses on obstacle negotiation, path planning, and position estimation. These issues have conventionally been the foremost factors limiting the performance and speeds of mobile robots. Less attention has been paid to date to the issue of terrain classification, which aims at associating terrain with well-defined categories, such as gravel, sand, or tile.

A related but different type of terrain analysis is terrain *characterization*, that is, determining characteristics of the terrain that affect the driving performance and safety of small mobile robots traversing the terrain. Yet, trafficability is of great importance if mobile robots are to reach speeds that wheeled-driven robots can reach on rugged terrain. For example, it is obvious that the maximal allowable speed for a turn is lower when driving over sand or wet grass than when driving on packed dirt or tile. In order to emphasize that the characterization methods discussed in this work relate to the trafficability characterization of the terrain, we use the term “trafficability” throughout this thesis.

[1]**Proprioception**: the perception by an animal of stimuli relating to its own position, posture, equilibrium, or internal condition. (Encyclopædia Britannica)

[2]**Exteroceptive**: activated by, relating to, or being stimuli received by an organism from outside.(Merriam-Webster Dictionary)

The main difference between characterization and classification is that characterization tells us how the terrain affects driving behavior (e.g., slippery and soft) without attempting to identify the terrain. Terrain classification, on the other hand, does not necessarily tell us how the terrain affects driving behavior, but it tells us what type of terrain it is.

An example for the significance of this distinction is this: In order for a legged robot to walk safely but at the highest possible speed over a tiled road, it is very important to know whether the tile is wet or dry. It is less important to know that the terrain is made of tile, as long as the robot knows what cornering forces or break distance the terrain will support. In contrast, a remote human operator may want to know if the robot is still driving over tiles or if it has reached a gravel–covered patch adjacent to a targeted position. In this example, terrain classification can help pinpoint the location of the robot on, say, an aerial map that the operator is using.

We selected the particular field of autonomous terrain identification as:

- We believe that robust terrain identification is a vital capability that mobile robots must possess. This is particularly true if these systems are to be deployed in natural, hostile or unknown environments. Therefore, the robot must have the ability to distinguish these terrain types to be able to avoid not only obstacle but also parts of the terrain that are considered not safe for navigation or not adequate to traverse. In such cases, the safety of the robot might depend on the nature of the immediate terrain. The locomotion behavior of the robot is also intimately tied to the terrain type. Thus, its identification in an efficient manner is very useful.

- In natural environments, terrains generally present some form of spatial coherence, even in unstructured environments. Also, environments adapted by humans (sidewalks, grassed area and roads) are patchy by design. This spatial coherence will be a key element in associating the identified local terrain characteristics with the remote predicted ones.

- Finally, it is our belief that the role of tactile perception in mobile robotics, outside of manipulation, has been so far under-exploited by the scientific community. Our work on terrain identification, as well as the use of this information to modify the behavior of a mobile robot, represent contributions to the applicability of tactile perception in mobile robotics.

This work aims at establishing an architecture from which the robot can classify and identify the type of terrain under its feet and relatively away from it. The terrain classification is done over the sensor data collected during its interaction with the terrain it encounters. We introduce a robot–terrain interaction metric that qualify this interaction.

The information characterizing the terrains, obtained from proprioceptive data (from local sensors, e.g., an inertial measurement unit) and exteroceptive data (from remote sensors, e.g., a camera) are used to characterize the respective metric. Which can be described as an estimation of the easy of traversability over a specific class of terrain, and machine learning algorithms are used to classify each of those classes. Subsequently, data acquired on the same terrain is used to associate the corresponding models with the same remotely inferred one.

In a second step, based on the remote sensor data measured, the robot–terrain interaction model is predicted using the inference model. This process corresponds to a near-to-far approach and provides the most probable metrics of the terrain lying ahead of the robot. The predicted metrics are then used to plan an optimal path with respect to the terrain–interaction model and therefore influence the robot trajectory.

In the context of this research, the CREX robot is used for the implementation and testing of the approach. We call this approach: "Intelligent Architecture for Legged Robot Terrain Classification Using Proprioceptive and Exteroceptive Data (*i*ARTEC)".

This thesis presents *i*ARTEC , describes its implementation, and concludes with results from simulation, experiments, and field tests that validate its subsystems.

1.2 Contributions

The goal of this thesis is to develop an intuitive, adaptive, and flexible architecture for legged robots to assert the robot terrain type and interaction forces based on the robot own proprioceptive and exteroceptive data and the robot-terrain interaction learned from experiments. This architecture is a hybrid architecture that combines physical modeling, machine learning, and deliberative control. Within the proposed architecture, we implement some new subsystems (for terrain classification and path planning and following) and integrates them with already available subsystems to accomplish its desired tasks.

Within this architecture, experiments with the robot leg–ground interaction with certain types of terrains are used to build a physical model to estimate the interaction forces with the terrain underfoot. In a following step, different proprioceptive data from the robot sensors are used with machine learning techniques to predict terrain class over which the robot is moving. Thus, based on a near-to-far approach, the interaction behavior can be estimated and abstracted into classes. The exteroceptive data from an attached camera to the robot is used to predict the remote terrain classes through a feature-based visual classifier. The identified terrains are used to plan a desired path for the robot to favor a "*better*" terrain to traverse.

The individual parts and the combination of methods form the contributions of the proposed architecture to the state of the art. The following is a list of technical contributions arising from this thesis:

- development of a flexible, adaptive, and intelligent architecture for legged robot terrain classification using proprioceptive and exteroceptive Data (*i*ARTEC).
- A model-based terrain characterization approach using proprioceptive sensors in that separates the process of terrain force estimation from model-based terrain characterization so that no strong assumptions regarding stress distributions at

the foot-ground contact patch are required and no terrain parameters need to be known a priori. This approach characterizes the terrain "underfoot" and therefore do not rely on vision sensors.

- A supervised machine learning approach to terrain classification using the robot embodied data measurements. The learning framework preprocesses the robot proprioceptive data through channel reduction and filtering such that the classifier is trained on the feature vectors that are closely associated with terrain class.
  - Methodologies, as outlined above, which characterize the terrain "underfoot" and therefore do not rely on vision sensors.
- development of a feature-based visual classifier to identify long-range terrain types in images with homogeneous and heterogeneous terrain patches.
- implementation of a terrain dependent navigation and path planning approach that is based on a proposed metric.
- design and implementation of a path follow controller for the robot to track the previously planned path.
- experimental test and validation of the presented approach over:
  - different sensing modalities (inertial, actuator, camera, ...);
  - different sets of real experiments and simulations;
  - various experiments to optimize the parameters and attributes of the subsystems.

1.2.1 Publications

This thesis is supported by a number of peer-reviewed publications (one journal article, 17 conference papers, and two talks) for each of the main contributions and related to the previously mentioned goals of this thesis. These publications are grouped under this thesis main topics as:

Robot foot-ground contact

- Ahmed, M. and Yoo, Y.-H. (2010). Measurement and control of the contact forces between walking robot legs and its environment. In *Proceedings of the Joint 9th Asia-Pacific ISTVS Conference and Annual Meeting of Japanese Society for Terramechanics*, Sapporo, Japan. International Society for Terrain-Vehicle Systems, ISTVS2010
- Ahmed, M., Quack, L., Langosz, M., and Yoo, Y.-H. (2011). Development of a real and simulation testbed for legged robot soil interaction. In *International Conference of the International Society for Terrain-Vehicle Systems, (ISTVS-11)*, pages 110–116, Blacksburg, Virginia, USA. International Society for Terrain-Vehicle Systems, ISTVS2011

- Langosz, M., Ahmed, M., Quack, L., and Yoo, Y.-H. (2011). Modeling of leg soil interaction using genetic algorithms. In *International Conference of the International Society for Terrain-Vehicle Systems, (ISTVS-11)*, pages 110–116. International Society for Terrain-Vehicle Systems, ISTVS2011
- Yoo, Y.-H., Ahmed, M., Bartsch, S., and Kirchner, F. (2010a). Realistic simulation of extraterrestrial legged robot in trade-off between accuracy and simulation time. In *Proceeding of the 35th Annual Conference of the IEEE Industrial Electronics Society (IECON-2010)*, Glendale, AZ, USA
- Yüksel, M., Oekermann, C., Girault, B., and Ahmed, M. (2014). Using industrial actuators for rapid development of electric car applications. In *Proceedings of the 14th International Conference on New Actuators (ACTUATOR-14); 8th International Exhibition on Smart Actuators and Drive Systems*, Bremen, Germany. MESSE BREMEN, WFB Wirtschaftsförderung Bremen , Bremen
- Yüksel, M., Ahmed, M., Girault, B., Birnschein, T., and Kirchner, F. (2014). A framework for design, test, and validation of electric car modules. In Fischer-Wolfarth, J. and Meyer, G., editors, *Advanced Microsystems for Automotive Applications 2014*, Lecture Notes in Mobility, pages 245–254. Springer International Publishing

Short–range embodied terrain classification

- Yoo, Y.-H., Ahmed, M., Roemmermann, M., and Kirchner, F. (2009). A simulation-based design of extraterrestrial six-legged robot system. In *2009 35th Annual Conference of IEEE Industrial Electronics*, pages 2181–2186, Porto, Portugal. IEEE
- Ahmed, M., Yoo, Y.-H., and Kirchner, F. (2010b). A co-simulation framework for design, test and parameter optimization of robotic systems. In *ISR / ROBOTIK 2010. The joint conference of the 41st International Symposium on Robotics and the 6th German Conference on Robotics (ISR/ROBOTIK-2010), June 7-9, Munich, Germany*. ISR/ROBOTIK2010
- Yüksel, M., Ahmed, M., Girault, B., Birnschein, T., and Kirchner, F. (2014). A framework for design, test, and validation of electric car modules. In Fischer-Wolfarth, J. and Meyer, G., editors, *Advanced Microsystems for Automotive Applications 2014*, Lecture Notes in Mobility, pages 245–254. Springer International Publishing
- Birnschein, T., Kirchner, F., Ahmed, M., Yueksel, M., Yoo, Y.-H., Oekermann, C., Girault, B., Kroffke, S., and Gruenwald, D. (2014). Enhancing mobility using innovative technologies and highly flexible autonomous vehicles. In *18th International Forum on Advanced Microsystems for Automotive Applications (AMAA 2014): Smart Systems for Safe, Clean, and Automated Vehicles*, Berlin, Germany. accepted
- Yüksel, M., Oekermann, C., Girault, B., and Ahmed, M. (2014). Using industrial actuators for rapid development of electric car applications. In *Proceedings of*

the 14th International Conference on New Actuators (ACTUATOR-14); 8th International Exhibition on Smart Actuators and Drive Systems, Bremen, Germany. MESSE BREMEN, WFB Wirtschaftsförderung Bremen , Bremen

- Ahmed, M., Eich, M., and Bernhard, F. (2014a). Design and control of mira: a lightweight climbing robot for ship inspection. In *World Symposium on Mechatronics Engineering & Applied Physics (WSMEAP2014). International Conference on Mechatronics Engineering (ICME-2014)*, pages 58–62, Sousse, Tunisia. IEEE
- Ahmed, M., Oekermann, C., and Kirchner, F. (2014b). Cosimulation environment for mechanical design optimization with evolutionary algorithms. In Sammouda, R., editor, *World Symposium on Computer Applications & Research WSCAR' 2014, International Conference on Artificial Intelligence (ICAI' 2014)*, pages 21–26, Sousse, Tunisia. IEEE
- Ahmed, M., Ebrahim, M. A., Ramadand, H. S., and Becherif, M. (2015). Optimal genetic-sliding mode control of vsc-HVDC transmission systems. *Energy Procedia*, 69. International Conference on Technologies and Materials for Renewable Energy, Environment and Sustainability, TMREES15
- Sonsalla, R., Ahmed, M., Fritsche, M., Akpo, J. B., and Vögele, T. (2014). Scout rover applications for forward acqusition of soil and terrain data. In *Proceedings of European Planetary Science Congress (EPSC-2014)*, volume 9, Cascais, Portugal, Portugal. EPSC

Long-range visual terrain classification

- Ahmed, M. and Kirchner, F. (2014). Design and implementation of a long range visual terrain classifier for legged robots. In Sammouda, R., editor, *World Symposium on Computer Applications & Research WSCAR' 2014, International Conference on Signal Processing and Remote Sensing (ICSPRS2014)*, pages 117–122, Sousse, Tunisia. IEEE

Path planning and following

- Ahmed, M., Sonsalla, R., and Kirchner, F. (2014c). Autonomous path tracking steering controller for extraterrestrial terrain exploration rover. In *40th COSPAR Scientific Assembly*, Moscow, Russian Federation. cosmos
- Ahmed, M. and Babu, A. (2014). Autonomous steering controller for path following. In *Proceedings of the RIC Project Day Workgroups "Framework & Standardization" and "Manipulation & Control". RIC Project Day, June 19, Bremen, Germany*, volume 14-05 of *DFKI Documents, D*, pages 118–119. DFKI Robotics Innovation Center Bremen, Selbstverlag

- Ahmed, M. and Kirchner, F. (2009). A simulation environment to be utilised in the design and test process of the HEVs and EVs BLDC drive and its control. In *11th European Regional Conference of the International Society of Terrain-Vehicle Systems (ISTVS-09)*, Bremen, Germany. ISTVS'09
- Ahmed, M., Benitez, L. V., and Kirchner, F. (2010a). Accurate identification and simulation of brushless DC drive actuating system for high performance applications. In *The 4th Int'l Industrial Control & Automation Technology Exhibition and Conference. Automation Technology Egypt 2010 (Automation -2010), May 10-12, Cairo, Egypt*. Automation 2010
- Ahmed, M. and Yüksel, M. (2013). Autonomous path tracking steering controller for EO smart connecting car. In Tagoug, N., editor, *Proceeding of the World Congress on Multimedia and Computer Science 2013. International Conference on Intelligent Automation and Robotics (ICIAR-13)*, pages 45–50, Hammamet, Tunisia. IEEE

Two talks

- Ahmed, M. and Babu, A. (2014). Autonomous steering controller for path following. In *Proceedings of the RIC Project Day Workgroups "Framework & Standardization" and "Manipulation & Control". RIC Project Day, June 19, Bremen, Germany*, volume 14-05 of *DFKI Documents, D*, pages 118–119. DFKI Robotics Innovation Center Bremen, Selbstverlag
- Ahmed, M. (2014). Car of the future: Innovative technologies in electromobility for development, design, and construction of smart cars. Talk presented at HANNOVER MESSE 2014, Forum Robotics, Automation & Vision

1.3 Thesis Outline

The remainder of this thesis is structured in eight chapters as outlined in Figure 1.1. As illustrated, the main contributions are presented in chapters four through seven. Chapters two and three are introductory and serve as a basis for developing the proposed architecture over several chapters of this thesis. Experiments test and verify the performance the different components of the proposed architecture are given in chapter eight. The last chapter summarizes the conclusions and future work. The following is a brief description of the

Chapter two discusses the related work on the use of tactile information in mobile robotics and terrain identification for wheeled and legged robots. This chapter also reviews various sensing techniques for terrain assessment.

Chapter three is a short chapter that introduces the architecture proposed in the thesis and its main subcomponents for the intelligent control of the interaction of robot manipulators. This chapter focuses on giving an overview of the approach, including a description of all the components needed to achieve it. This description is very generic and it does not take a specific robotics platform into account. The second part of this chapter describes the CREX robot platform, which is a six motorized legged robot used for most of the work described in this dissertation. The mechanics, electronics and low-level control are presented in some details as the robot was used for most of the implementations and test aspects of our work.

Chapter four describes some of the analysis methods of robot foot–ground interaction forces and develops the utilized plastic terramechanics particle model.

Chapter five presents machine learning techniques related to our work, specifically support–victor machine techniques exploiting temporal coherence. This will be followed by a description of the machine learning tool used. The details of the short-range terrain classifier will also be given here.

Chapter six presents the long–range visual terrain classifier and its parameters, attributes , and design.

Chapter seven develops a high-level terrain type dependent path planning approach and the development of the low-level path following controller.

Chapter eight presents the experiments performed to evaluate the architecture and their results are described in detail. Four groups of experiments describe different aspects of the proposed architecture, from the learning to the overall approach, using different terrains and setups. The results presented show the interest and impact of the whole research. Finally,

Chapter nine concludes this thesis with the final conclusions summarizing the outcomes and contributions of the research. An outlook showing the possible future development is also included.

Figure 1.1 shows the structure of the thesis grouped into its main topics.

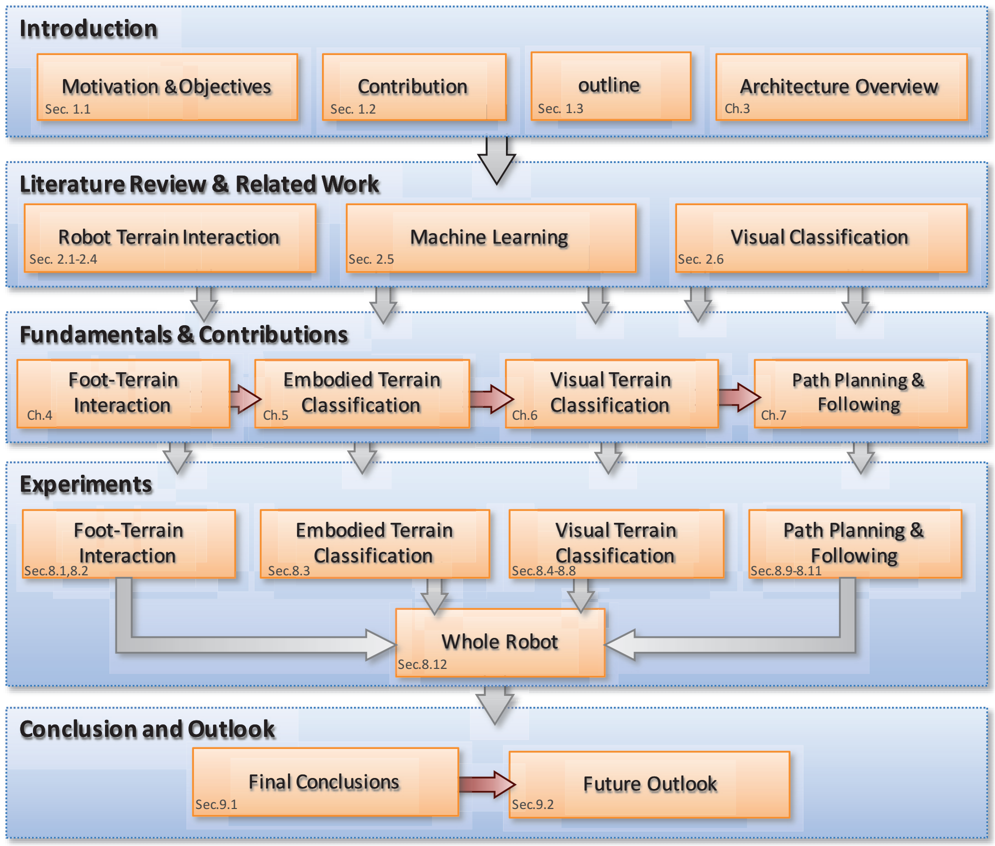


Figure 1.1: Thesis outline. The numbering is used for the respective chapters and sections.

Disclaimer: Text Reuse

The material presented in this thesis has not previously been submitted for a degree to any University. Parts and sections of this thesis have been previously published in a number of publications which were by me as the main author or coauthor. The parts included in this thesis are taken from my own publications without explicit quotation because I am the main author: [Ahmed et al., 2014a],[Ahmed et al., 2014b],[Ahmed et al., 2015], [Ahmed and Kirchner, 2014],[Ahmed et al., 2014c],[Ahmed and Yüksel, 2013], [Ahmed et al., 2011],[Ahmed et al., 2010b],[Ahmed et al., 2010a], [Ahmed and Yoo, 2010],[Ahmed and Kirchner, 2009],
or I contributed the used part to them: [Langosz et al., 2011, Sonsalla et al., 2014, Yüksel et al., 2014, Yüksel et al., 2014, Birnschein et al., 2014, Yoo et al., 2010a, Yoo et al., 2009], which is somehow scattered over this dissertation, I explicitly mention these sources in the respective chapters or sections where they are used. The respective parts are no complete copy of the original papers. Often parts of these papers could be omitted by referring to other sections. On the other hand, additional information, additional experiments, the relation to the other parts of this thesis, or personal experiences are added.
There have been, however, a number of components, algorithms, and/or modules used in this work that are made by people outside of myself and my supervisor. An example of such is the robot CREX. It is the work of a number of people over many years within the [TransTerrA Project, 2015]. Explicit references of the sources of these materials are given in the respective chapters or sections where they are used.

Notation

In this thesis mostly the "standard notation" is used and it should be possible to infer the meaning from the context. If in some circumstance the standard symbols are not used, it will be directly mentioned. Nevertheless, there is a list of acronyms and a list of used symbols at the end of this document. If some notation is unclear refer to these lists.

1.4 Summary

This chapter serves as introduction to this work and describes its motivation, scope, and goals. Moreover, the chapter shows the structure of this document and summarizes the most relevant contributions of this work to the state of the art.

"It is impossible for us, who live in the latter ages of the world, to make observations in criticism, morality, or in any art or science, which have not been touched upon by others. We have little else left us but to represent the common sense of mankind in more strong, more beautiful, or more uncommon lights."

— Joseph Addison, (English essayist, poet, and politician, 1672–1719), Spectator, No. 253

2 State of the Art

The work presented in this dissertation is founded on and related to numerous fields, notably mobile robotics, legged robots, machine learning, optimization and computer vision and image processing. Since it would be impossible to do a proper review of all these fields, the review of previous literature given in this chapter will concentrate around three center areas related to this work:

- The first area pertains to sensors and sensor interpretation in the context of terrain identification. Therefore, we do an overview of what has been done for terrain characterization in terms of soil properties or geometry in Section 2.1. In this section, we give an account of outdoor wheeled vehicles. Then we present more work on terrain identification, but targeted towards legged robots, which is the chosen platform for our research.
- The second area of this review is related to machine learning and optimization techniques. Since a major aspect of our architecture is the exploitation of the sensor data for classification, we discuss similar classification techniques in Section 2.3.
- Finally, in Section 2.2, we go over the visual-based approaches which is another core component of our research. These approaches use mainly computer vision and image processing techniques to classify the terrains by recognizing or identifying the terrain type on which the robot is driving. We begin by presenting the researches aiming at asserting the traversability of the terrain. Then we describe the approaches allowing to classify the terrains through different image classification.

Although a relatively new field, autonomous terrain assessment has been studied and applied to a variety of different robotic platforms (e.g. wheeled, tracked robots and rovers) and applications (e.g. hazard avoidance, soil properties estimation, automated detection of scientifically interesting formations). The following sections aims to present a comprehensive and organized overview of the field.

The chart in Figure 2.1 depicts the main research trends related to this work as trying to identify the terrain around the robot. A short description of the methods with advantages and disadvantages is given. The inter-relations between these approaches are also illustrated.

2.1 Terrain Characterization

Terrain characterization has been the subject of several studies, although these earlier studies were typically aimed at larger vehicles and the characterization process was manual. Perhaps the best known and widely cited works are those of [Bekker, 1960] and [Wong, 2014]. From the point of view of terramechanics, soil can be characterized by determining the terrain parameters. Many approaches to terrain characterization require offline analysis and/or dedicated equipment [Nohse et al., 1991; Shmulevich, Ronai & Wolf, 1996]. Terrain characterization without dedicated equipment was proposed for the Sojourner rover and its 1997 Mars mission [Matijevic et al., 1997]. Based on this method, Sojourner used one of its wheels to characterize terrain.

A real–time approach based on the measurement of wheel sinkage in soft soil using a video camera was proposed in [Iagnemmaet al., 2004]. In this paper, they proposed a fully self-contained terrain characterization method for skid-steer mobile robots. They defined “self-contained” as system does not require any special-purpose instruments to be attached to the robot. Rather, the proposed method monitors typical onboard sensors, such as gyros and motor current sensors. The unique advantage of this approach is that it can be applied during real time and during an actual robot mission. In numerous runs, they collected data on five different terrains: gravel, sand, tile, grass, and dirt. Sensor data were collected while the robot performed carefully prescribed maneuvers. The resulting data were analyzed with the proposed method, which yielded a curve that is characteristic for a particular terrain.

For rolling systems, the interaction between the robots and the terrain occurs at the level of the wheels. The study of the soil properties in this context, called *terramechanics*, had been studied by [Bekker, 1956; Bekker,1969] and more recently [Wong, 2001][Wong, 2014]. The equations modeling the wheel-ground interaction are involving a great deal of parameters, which are difficult to estimate in the context of exploration robotics. [Iagnemma et al., 2004] presented a method of online evaluation of those parameters for a single wheel rolling over sandy surfaces. Knowing these parameters allows to enhance the robot control, as [Ishigami et al., 2006] which proposed a slippage model in relation with the terrain slope. In a different context, [Heimann and Blume, 2006] addressed also the modeling of wheel-ground interaction and presented a method to estimate the friction coefficient for cars.

These models are interesting and well designed, but they rely on a huge number of parameters that are hard to define in a real robot exploration mission. Besides characterizing the soil parameters, other researchers have taken into consideration the geometry of the terrain. Thus [Lacroix et al., 2002] and [Ingrand et al., 2007] generated a 3D representation of the environment and the robot can adapt its control with respect to the elevation map of the terrain. Such maps are widely used and other work focus on specific sensors and environment, such as [Williams and Howard, 2009] for the arctic.

In the last decade the desire to increase the off-road navigation capabilities of unmanned ground vehicles (UGVs) has led to an increasing number of research activities in the field of autonomous terrain assessment. legged robots are a particularly challenging type of UGVs designed to operate in all-terrain purposes seeing legs as more effective in an uneven environment than wheels. Furthermore, they need to traverse long distances on uncharted natural terrain with a minimum of human intervention. The primary objective of most legged robots is scientific and therefore most engineering efforts are focused on increasing and improving the autonomy level of the robot. Higher autonomy enables the robot to traverse longer distances. In order to navigate autonomously, legged robots need to assess the nearby terrain, avoiding obstacles and other hazards such as excessive tilt and roughness. Ideally, they should also steer preemptively away from undesirable areas, favoring kinematically and dynamically feasible paths through safer soils.

The knowledge of certain terrain properties can also be used to correct trajectory deviations and increase the stability of the robot, applying techniques such as slip compensation or traction control. In the near future, legged robots are expected to be able to assess the scientific interest of different geological formations, extending the navigation aspects of terrain assessment to mission planning and scientific data selection and labeling.

2.1.1 Terrain Characterization for Wheeled Vehicles

Several techniques have been developed for terrain characterization [Weiss et al., 2006, Sadhukhan and Moore, 2003, Brooks and Iagnemma, 2005, Dupont et al., 2008]. Features are extracted from acceleration measurements (i.e., frequency spectrum of acceleration, multiple moments, threshold crossings, etc) and supervised learning is used to train a classifier, such as a support vector machine [Weiss et al., 2006] or a probabilistic neural network [Dupont et al., 2008]. In some other work [Giguère and Dudek, 2009] a non-parametric classifier for time-series is trained to identify states of a legged robot interacting with its environment. These techniques require part of the data to be manually labeled.

One of the first demonstrations of terrain identification for wheeled vehicles using on-board sensors is due to [Sadhukhan and Moore, 2003]. They relied on the measured vertical acceleration of the chassis of a wheeled vehicle to perform terrain identification. The features used for classification were the amplitude spectrum of the acceleration, within a frequency window going from 10 to 20 Hz. A Probabilistic Neural Network was trained to differentiate between three different surfaces: gravel, packed dirt and grass. An interesting conclusion of their research was that the classification success rate for their method was increasing with the velocity of the vehicle, going from 75% at 0.2 m/s to 94.7% at 0.8 m/s. They explained this phenomenon by the fact that, at higher velocities, the vibration patterns between surfaces become increasingly different.

[Brooks and Iagnemma, 2005] developed an algorithm for future planetary explorations where rovers traverse very rough terrain with limited human supervision. They agree that terrain estimation will increase the possibility of a successful mission by enabling a rover to adapt its control and planning strategies. The goal of the algorithm is to determine the soil shear strength from two key terrain parameters: cohesion of the soil and internal friction angle. In [Brooks and Iagnemma, 2007], these parameters are estimated online using a simplified form of classical terramechanics equations. Finally using Coulomb's equation, these two parameters can be combined to give the shear strength of the soil. This algorithm has limitations related to sensor noise. Also, it does not explicitly determine the type of terrain.

[DuPont et al., 2006] expanded the sensory modalities used for identification. They added two angular velocities, on top of vertical acceleration, to improve the identification success rate. These angular velocities were measured by rate gyroscopes, oriented in such a way as to be sensitive to the pitching and rolling motions of the chassis. The respective power spectra were computed and PCA was applied on the spectra to reduce the correlation between sensing modalities. The final dimensionality of the problem was reduced to 25 down from an original 45. The best classification results for gravel, grass and asphalt were 97.8% at 0.8 m/s and 80.0% at 1.2 m/s. In [Dupont et al., 2008], no dimensionality reduction was done. The number of tested surfaces was increased to six and consisted of packed gravel, loose gravel, sparse grass, tall grass, asphalt and sand. The average classification success rate was 94.4% at 0.5 m/s and 92.2% at 1.0 m/s.

[Ojeda et al., 2006] performed terrain classification for five ground types: gravel, grass, sand, pavement, and dirt. They experimented with a variety of different sensors, including microphones, gyroscopes, accelerometers, encoders, motor current, voltage sensors, and downward-facing ultrasonics and infrared. They used a separate classifier for each sensing modality and analyzed which modalities worked best for specific terrains, but did not attempt to combine the results. They found that gyroscopes worked best for gravel, pavement and dirt; motor current worked best for sand; and microphones worked best for grass.

The approach of [Weiss et al., 2006] is similarly based on analyzing the vibration patterns of an accelerometer mounted on the body of a wheeled cart. They extracted features from the z-value of an accelerometer that is part of an Attitude Heading and Reference System (AHRS). The AHRS was strapped to a wheeled cart that was manually pushed over different terrain types, with the intent to later collect these signals from a mobile robot.

They extracted eight features from 1 second time windows for classification, without any further dimensionality reduction. These features were, for the most part, signal statistics taken in the time domain, with the exception of one autocorrelation score. A support vector machine (SVM) was trained for classification. Test sets were gathered over seven different terrain types: vinyl floor, asphalt, gravel, grass, paving, Boule court (clay), and no motion. The overall classification success rate for these seven surfaces was 94.7 percent. A comparison of some of the different approaches based on vibration to used for terrain classification can be found in [Weiss et al., 2006].

Coyle and Collins also performed vibration-based classification to distinguish between seven ground types: beach sand, packed clay, regular grass, tall grass, loose gravel, packed gravel, and asphalt [Coyle et al., 2008]. They used the z-value, roll rate, pitch rate, and speed reported by an Inertial Measurement Unit (IMU) on-board a mobile robot.

Stavens *et al.* used imitation learning to model a velocity controller for a high-speed, off-road vehicle [Stavens et al., 2007]. Shock measurements from an IMU as well as geometric terrain features from LADAR were used to model the roughness of the ground. A human drove through some example terrains, and then imitation learning was used to understand the ways in which the human accelerates on different ground types. Stanford's 2005 DARPA grand challenge platform used this algorithm for some of its speed control.

In [Brooks and Iagnemma, 2007], a terrain identification targeted at planetary rovers is presented. In their set-up, a vibration sensor mounted near the wheel of the rover recorded accelerations as the wheel rolled on various surfaces. The power spectrum logarithm for 1 seconds time windows of acceleration measurements was used as classification features. PCA was employed to reduce the dimensionality to an empirically-chosen value of 15.

Classification was performed using a pairwise linear classifier trained using Linear Discriminant Analysis. For cases involving more than two terrains, a pairwise classifier was trained for every possible pair of terrains. In those cases, classification was performed using a voting scheme across all possible pairwise classifiers. When the voting results were confident (i.e. above a threshold), this technique had an average success rate of 91 percent in distinguishing between gravel, sand and Mars-1 soil simulant.

All of these terrain identification methods were based on the use of a classifier trained by supervised learning. This implied that a training data set was collected beforehand and had to be manually labeled. Consequently, these methods are not applicable when a training data set is unlabeled, as in the case of trying to learn to identify terrain autonomously.

Table 2.1 and Table 2.2 summarize the implementations discussed above for supervised learning on mobile robots that use proprioceptive sensing. Most of this body of work focused on classifying the material properties of the ground, except for [Stavens et al., 2007] who learned velocity control parameters.

Table 2.1: Comparison of terrain identification strategies for wheeled vehicles. The differences between techniques reflect the four basic components of intelligent classification systems: sensing modalities, feature extraction, dimensionality reduction and type of classifier.

Author(s)	Sensor	Location of Sensor	Extracted Features	Dimensionality Reduction	Classifier
Sadhukhan[a]	Accelerometer	Vehicle Body	$10 - 20$ Hz Amplitude Spectrum	-	Probabilistic Neural Network
Brooks & Iagnemma[b]	Accelerometer	Near Wheel Axle	Log of Power Spectrum	PCA	linear separator + LDA + voting
Weiss[c]	Accelerometer	Vehicle Body	Signal Statistics	-	SVM
DuPont[d]	Accelerometer + 2 rate gyros	Vehicle Body	3-30 Hz Amplitude Spectrum	PCA	Probabilistic Neural Network
DuPont[e]	Accelerometer + 2 rate gyros	Vehicle Body	3-30 Hz Amplitude Spectrum	none	Probabilistic Neural Network

[a] [Sadhukhan and Moore, 2003] [b] [Brooks and Iagnemma, 2005]
[c] [Weiss et al., 2006] [d] [DuPont et al., 2006] [e] [Dupont et al., 2008]

Table 2.2: Comparison of properties estimated from supervised learning algorithms that use proprioceptive sensors on mobile robots.

Author(s)	Properties Estimated
Stavens et al.[a]	Velocity control parameters
Ojeda et al.[b]	Gravel, grass, sand, pavement, dirt
Weiss et al.[c]	Vinyl floor, asphalt, gravel, grass, paving, clay court, no motion
Coyle & Collins[d]	Beach sand, packed clay, regular grass, tall grass, loose gravel, packed gravel, asphalt

[a] [Stavens et al., 2007] [b][Ojeda et al., 2006]
[c] [Weiss et al., 2006] [d][Collins and Coyle, 2008, Coyle et al., 2008]

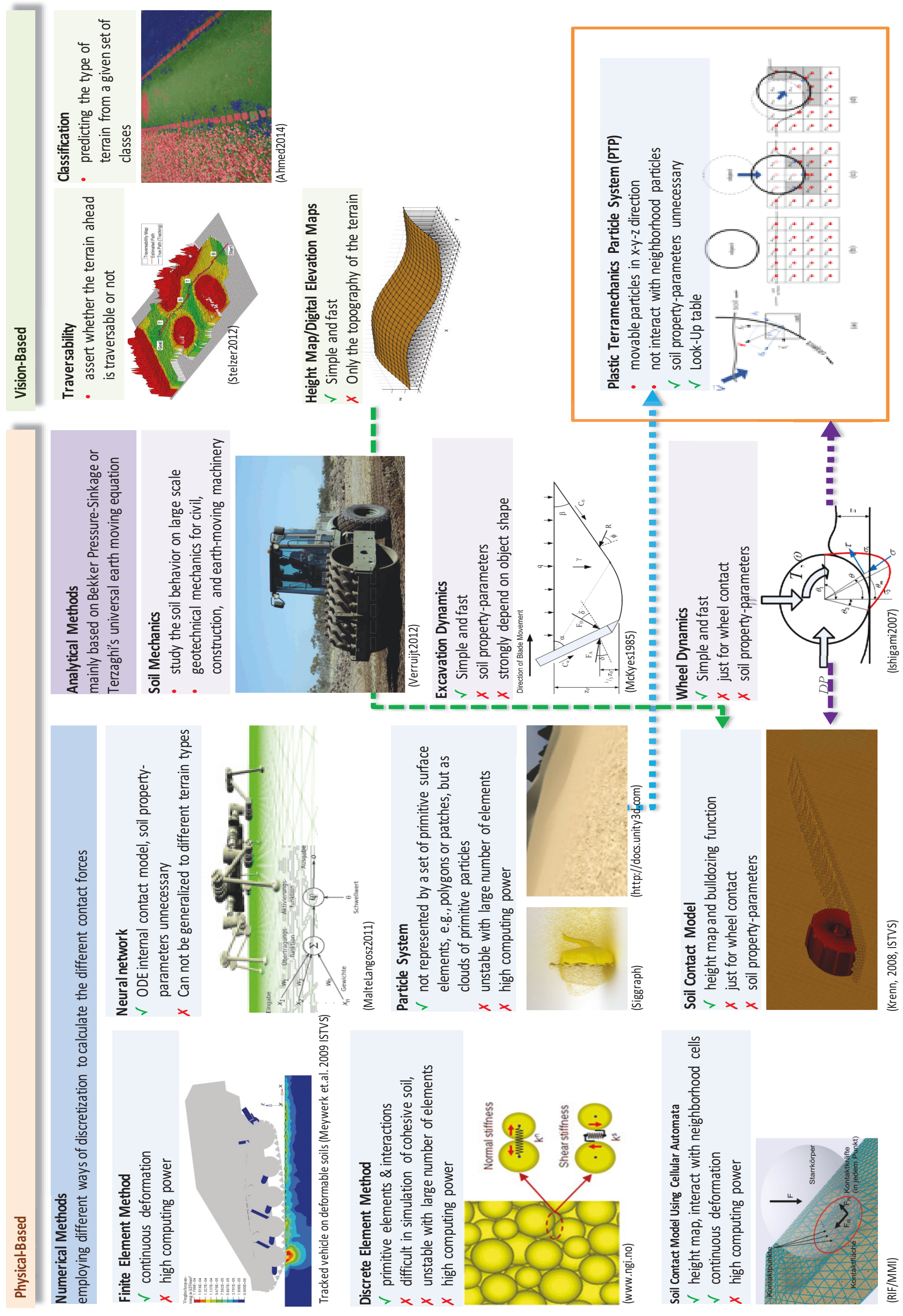

Physical-Based
Vision-Based
Numerical Methods
employing different ways of discretization to calculate the different contact forces
Finite Element Method
✓ continuous deformation
✗ high computing power
Tracked vehicle on deformable soils (Meywerk et.al. 2009 ISTVS)
Neural network
✓ ODE internal contact model, soil property-parameters unnecessary
✗ Can not be generalized to different terrain types
(MalteLangosz2011)
Discrete Element Method
✓ primitive elements & interactions
✗ difficult in simulation of cohesive soil,
✗ unstable with large number of elements
✗ high computing power
Normal stiffness
Shear stiffness
(www.ngi.no)
Particle System
✓ not represented by a set of primitive surface elements, e.g., polygons or patches, but as clouds of primitive particles
✗ unstable with large number of elements
✗ high computing power
(Siggraph)
(http://docs.unity3d.com)
Soil Contact Model Using Cellular Automata
✓ height map, interact with neighborhood cells
✓ continuous deformation
✗ high computing power
(RIF/MMI)
Soil Contact Model
✓ height map and bulldozing function
✗ just for wheel contact
✗ soil property-parameters
(Krenn, 2008, ISTVS)
Analytical Methods
mainly based on Bekker Pressure-Sinkage or Terzaghi's universal earth moving equation
Soil Mechanics
• study the soil behavior on large scale
• geotechnical mechanics for civil, construction, and earth-moving machinery
(Verruijt2012)
Excavation Dynamics
✓ Simple and fast
✗ soil property-parameters
✗ strongly depend on object shape
(McKyes1985)
Wheel Dynamics
✓ Simple and fast
✗ just for wheel contact
✗ soil property-parameters
(Ishigami2007)
Traversability
• assert whether the terrain ahead is traversable or not
(Stelzer2012)
Classification
• predicting the type of terrain from a given set of classes
(Ahmed2014)
Height Map/Digital Elevation Maps
✓ Simple and fast
✗ Only the topography of the terrain
Plastic Terramechanics Particle System (PTP)
• movable particles in x-y-z direction
• not interact with neighborhood particles
✓ soil property-parameters unnecessary
✓ Look-Up table

Figure 2.1: Chart for terrain identification state of the art related to this work. The approaches "Plastic Terramechanics Particle System (PTP)" and vision based terrain classification are presented in Chapter 4 and Chapter 5, respectively.

2.1.2 Examples of Prominent Terrain Identification Systems

NASA JPL Lunar Surface Operations Simulator (LSOS)

LSOS [Nayar et al., 2008, Nayar et al., 2009] has been built on and extended from previous simulation packages developed at the Jet Propulsion Laboratory (JPL). The primary model components in LSOS that enable the lunar mission simulations are terrain and rover systems. In addition to dynamic simulation of mechanical components like vehicles driving over terrain, LSOS implements process dynamic models. These include power generation, storage and usage, communication signal strength, temperature and other processes that track variables of interest. Within LSOS, there are two long traverse types: a 3 km field-trial traverse at Black Point Lava Flow in Arizona and a 570 km traverse from Shackleton Crater to Malapert Mountain and back to Shackleton Crater near the South Pole of the moon, were simulated in LSOS.

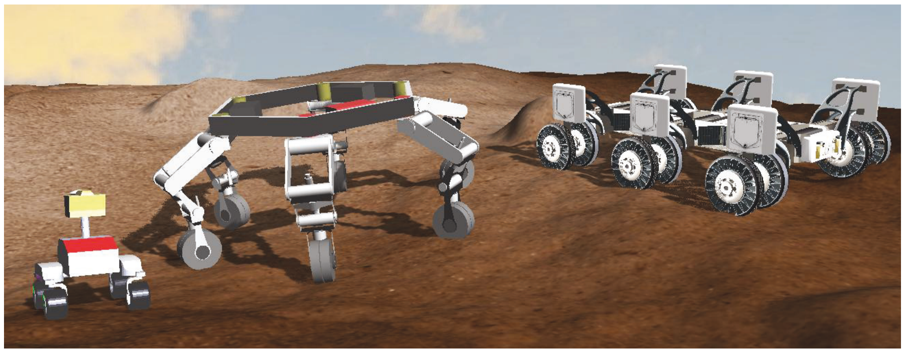


Figure 2.2: The Lunar Surface Operations Simulator with terrain and different robot models. Shown robots are: K-10 (left), ATHLETE (middle) and Chariot (right) rovers modeled in LSOS [Nayar et al., 2008].

Surrey Space Center Soil Mechanics Model

Surrey Space Centre (SSC) at the University of Surrey introduced a soil trafficability model based [Scott and M., 2009] for legged vehicles on granular soils. This model incorporates a validated draught force model based on Grisso's work, a validated active force model proposed by Bekker [Bekker, 1960], and a soil thrust model developed by Terzaghi [Terzaghi, 1943][Yeomans et al., 2013]. All experiments with this model, however, have been limited to the behaviors with very slow (5-6 seconds) and a simple pattern. A MATLAB based software tool, called Legged Performance and Traction Prediction Tool (LPTPT), with the soil-contact model is used for the simulation and any results of real-time capability are not reported. An introduction to the planetary soil simulants used in this study (SSC-1 quartz-based sand and SSC-2 garnet-based sand) and the process used to determine their mechanical properties is presented.

DLR Soft Soil Contact Model (SCM)

Soil Contact Model (SCM) [Krenn and Hirzinger, 2009] has been developed at DLR's Institute of Robotics and Mechatronics to provide an interface between the classical Bekker terramechanics theory and the capabilities of multi-body system simulation technique for general full 3D simulations of soil contact dynamics problems. This model is mainly used for rover mobility performance simulation on a sandy terrain and no results of real–time capability are reported. The SCM considers typical terramechanical phenomena like rising rolling resistance caused by humps in front, lateral guidance inside ruts, drawbar pull variations versus slippage and multi-pass effects of wheels running inline. SCM is available within SIMPACK as a tool for all design phases of planetary rover chassis development.

2.1.3 Soil Mechanics

It is noteworthy to mention here that in soil mechanics the main objective is also terrain assessment. As soil mechanics is the branch of civil engineering concerned with the engineering behavior of earth materials and its geotechnical mechanics for civil, construction, and large earth-moving machinery. The approaches adopted are to study the soil behavior on the large scale. In addition, soil classification in this scope is the geotechnical classification of the soil particle types. This provides information about the characteristics of the soil grains themselves (e.g. their grain size distribution) [Verruijt, 2012]. For these reasons, we see that it is not suitable to be used for mobile robots.

2.1.4 Physical-Based Numerical Methods

Finite element models (FEM) are particularly suitable to investigate complex problems involving geometric and material nonlinearities. [Nakashima et al., 1990], and [Liu and Wong, 1996] have used FEM to investigate soil-tire/track investigation using rigid-plastic model and critical state soil constitutive equation, restrictively.

[Yong et al., 1980] used Hertz contact theory to predict the pressure distribution between the soil and the wheel for use as a boundary condition in their FEM model. They also incorporated visco-plasticity effect into their FEM model. In [Schmid, 1995] a FEM soil-elastic robot interaction model (VENUS) was developed. The soil was characterized as an elasto-plastic material subject to the Drucker-Prager yield criterion. The robot was modeled as three homogeneous, elastic, concentric rings (tread, carcass, and wheel-rim).

[Nakashima and Oida, 2004] have proposed an interesting approach in which they combined discrete (distinct) element method (DEM) and FEM to solve the soil-robot interaction problem. [Zang and Zhao, 2013] also proposed an approach that combines the 3D discrete element method (DEM) and finite element method (FEM) together to investigate the running behavior of rigid wheel on sand terrain.

A related field to FEM and DEM analysis methods is the Cellular Automata (CA). An example of this method can be found in [Pla-Castells et al., 2006] where they present an application of CA in the field of dry Granular Systems (such as dry sand) modeling. A model for the reaction to external forces and a pressure distribution model between sand and spherical object is modeled and simulated.

In spite of the promising advances made in numerically predicting soil–robot interaction based on these methods. These methods do not have a widespread use for robot (wheeled and legged) terrain interaction mainly because they are very computationally expensive. From a computational point of view, no efficient models exist due to complexity of the problem and need for numerous soil parameters and also inadequate knowledge of boundary conditions.

2.2 Visual-Based Terrain Identification

The visual-based terrain identification methods project the problem into the image processing classification domain. In this perspective, the robotic systems employing such methods usually predict the terrain type from a discrete set of terrain classes.

The techniques under this approach can be decomposed into two major categories, namely, geometry-based and appearance-based. The following sections review these categories.

2.2.1 Traversability

Traversability estimation is one of the geometry-based methods that is trying to assert whether the terrain ahead is traversable or not, before negotiating with it. These remote based methods use different combinations or types of remote sensors and most of them making use of near-to-far approaches.

Lalonde et al. (2006) and Vandapel et al. (2004) make use of 3D Light Detection And Ranging (LIDAR), Manduchi et al. (2005) uses a combination of a stereo camera and a 2D LIDAR, Kim et al. (2008) and Poppinga et al. (2008) a 3D LIDAR and a camera, whereas Sofman et al. (2006) associates 3D LIDAR data and bird-eye camera views. All these approaches aim at asserting the traversability of the terrain, information then used to plan the robot's route toward the goal and to ensure its safety.

Since 2005, the DARPA-funded Learning Applied to Ground Robots (LAGR) project (JFR 2006, Mulligan and Grudic 2006) contributed enormously to the development and integration of these approaches. It offered a common robotic platform to develop and test off-road algorithms and compare the results. Trials were regularly organized, offering challenging environment in which the robot had to find its way to reach the goal, and learn from its experiments. Eight teams took part in this competition and the results are presented in Jackel et al. (2007). Among the best performing teams, Bajracharya et al. (2009) from Jet Propulsion Laboratory (JPL), Konolige et al. (2009) from Stanford Research Institute (SRI), Kim et al. (2006; 2007) from Georgia Tech, and Sermanet et al. (2009) from New-York University, give an overview of the approaches and techniques developed by their respective teams. For all the teams, the results showed the advantages of enabling a robot to learn from its experiment to improve its behavior in future, similar situations. The notion of near-to-far was introduced and used by several teams and it means that information learned at close range is used to learn to predict traversability at long range.

Nevertheless, these researches are focused on giving an answer to whether the terrain is traversable or not. This is expected as it is in line with the goals of the LAGR project which offers the robot several trials to reach a goal, enabling the robot to learn from its previous experiences. In this context, several vision based works, such as Hadsell et al. (2009) and Huang et al. (2009), made use of an image-based classification to enhance the traversability estimation of the terrain. Thus, from near-to-far, the approach becomes near-to-further, extending the perception horizon of the robot.

In a different context, Wellington et al. (2006) worked as well on the robot perception to assess the traversability. Using a tractor as robotic platform, the goal was to assert the true ground height in vegetation by making use of a near-to-far approach. This work showed impressive results, but this information only allows the autonomous vehicle to define whether the terrain is traversable or not.

2.2.2 Terrain Classification

In the context of this paper, “terrain classification” is the act of identifying the type of the terrain being traversed, from among a list of candidate terrains.

Early work on terrain classification specifically for robotics applications–focused on analyzing the texture of video images and synthetic aperture radar images. A survey of work performed with these two types of sensors can be found in Weszka, Dyer & Rosenfeld (1976) and Belhadj, Saad, El Assad, Saillard & Barba (1994), respectively. Another popular sensor modality for terrain classification is video; because video cameras are small, lightweight, emit no detectable energy, and they are inexpensive. Examples for work on wheeled mobile robots can be found in Talukder et al. (2002), and for legged robots in Larson, Voyles & Demir (2004). With the arrival of three-dimensional range sensors, researchers have also used range information for terrain classification purposes (Vandapel, Huber, Kapuria & Hebert, 2004).

Our proposed terrain classification system uses typically available on-board sensors, such as gyros, accelerometers, encoders, as well as motor current and voltage sensors. In addition, the paper describes a number of less commonly used sensors and their effectiveness with regard to terrain classification. These sensors are downward-facing ultrasonic and infrared sensors, as well as microphones. Signals from each sensor modality are fed into trained neural networks (NNs), one NN for each sensor modality. The output of each NN is a number between zero and one that indicates the likelihood of the present terrain being one of five previously defined and trained terrains. Somewhat related work has been proposed by [Sadhukhan and Moore, 2003] and [Brooks and Iagnemma, 2005, Brooks and Iagnemma, 2007]. However their work was limited to the use of accelerometers.

In parallel, other researches have considered to classify terrain with respect to a predefined set of categories and a preliminary training. Thus Ojeda et al. (2006) uses many sensory inputs to obtain the robot-terrain characteristics and classify them according to typical categories such as Grass, Gravel, Sand, Pavement and Dirt. Other researches use the same concept but focuses on a more specific robot-terrain characteristic.

In the case of Brooks and Iagnemma (2005; 2007) and Weiss et al. (2006; 2008), they make use of the vibration induced in the robot structure by the robot-terrain interaction. Vibration is mainly measured in terms of the linear acceleration of the wheel-bars or the robot body and Weiss et al. (2007) proposes a comparison of different methods for classification of terrains using a frequency-based representation. In relation with the vibration characteristics, the effect of the robot's velocity on the terrain classification is studied in Dupont et al. (2006; 2008) and a velocity independent classification method is proposed and tested on a very limited number of terrains. Similarly, Ward and Iagnemma (2009) proposes a speed independent solution for vehicles with a higher dynamics, classifying terrain using a single suspension mounted accelerometer.

Halatci et al. (2008) proposes a classification method making use of both visual and vibration data, comparing several classification methods. It is attractive, and mentioned here, as this approach mix both robot–terrain interaction and RTP data for the classification.

The methods presented here are interesting and show great results, but by defining a given number of terrain classes and training the classifiers before hand, these approaches are supervised and lack the flexibility a robot is needing in an exploration mission, while facing unknown areas.

2.3 Machine Learning

The learning mechanisms involved in mobile robotics are diverse in term of techniques or approaches. Thus Ollis et al. (2007) learns by imitation the correct behavior of a robot which is manually driven for a time. On a different level, the LAGR-related researches include learning approaches improving the robot behavior between the runs. Similarly, Wellington and Stentz (2004) enables its robotics platform to learn and adapt online the knowledge about the geometry of the terrain.

Such an online, near-to-far approach is appealing and it would be interesting to apply it to the robot–terrain interaction. Also note that for a more accurate description of the terrains encountered, and an increased flexibility, a novelty detection method which has the capability to add new terrain classes should be introduced.

Brooks and Iagnemma (2009) proposes a new novelty detection method and applies it to visual terrain representation. This process, based on SVM classifiers, is capable of detecting new terrain classes by having a priori knowledge about the known classes, and most importantly, about the "world" class. Therefore a preliminary input about the robot environment must be provided. Regarding the robot–terrain interaction models, Weiss and Zell (2008) agrees that autonomous system, in addition to learning from training data, should be able to detect and classify new terrains. The authors propose a really nice vibration-based classification method using a Gaussian mixture model for detection and classification of novel terrains.

Finally, Angelova et al. (2007) must also be presented in this section. In this work, another famous characteristic is used to express the robot-terrain behaviors; the slippage. This research proposes to learn the slippage model of the robot with respect to the terrain type and geometrical characteristics.

2.4 Summary

A sufficiently extended review of the state of the are and related work are provided that includes the most representative works and cover the major directions that are explored in this thesis. In this effort, a number of relevant works were omitted from the current review in order to give emphasis to those parts of the previous work that are deemed as maximally related to the topics of this dissertation and, in turn, minimize the redundancy in the referenced material at the benefit of the reader.

The different research approaches previously presented focus on characterizing the terrain, in terms of soil properties, terrain geometry, or terrain visual features. In the end, the goal is similar for most research types in the sense that the information obtained is used to optimize the robot control, or its global behavior. Which is a very appealing concept that we would like to introduce in our work.

"A computer would deserve to be called intelligent if it could deceive a human into believing that it was human."

— Alan Turing, (British pioneering computer scientist, cryptanalyst,⋯, and philosopher, 1912–1954)

3 Architecture Overview

This chapter aims to explain how the data is acquired and handled in the context of the architecture and the mechanisms involving the proprioceptive sensor data and long range visual terrain classification. Here also, an overview of the building blocks and the subsystems used to implementation the whole architecture are described.

The presented components in this work are combined with other subsystems of the robotic system and integrated within our proposed architecture as outlined in Figure 3.1

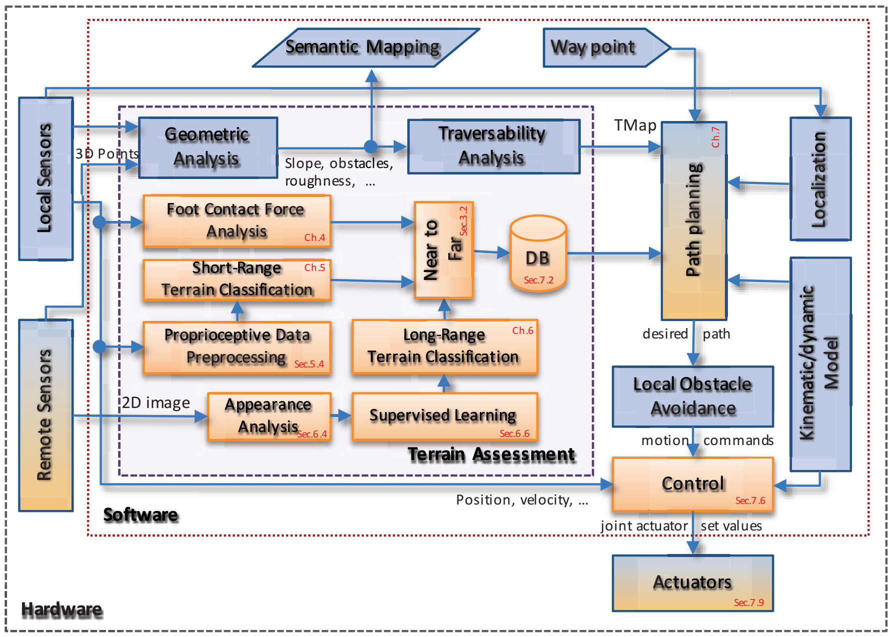


Figure 3.1: Functional configuration of a robot autonomous operation including terrain assessment with our architecture important modules and dataflow. The corresponding chapter and section numbers for the details of the respective components and subsystems are given.

3.1 Main Components of Terrain Identification in the Proposed Architecture

The terrain identification (characterization and classification) problem consists in categorizing a local and remote terrain based on sensor information. This identification problem, The main components of terrain identification in our architecture are shown in Figure 3.2. At the lower level, we have a collection of sensors measuring some of the terrain properties. Relevant features are extracted from the preprocessed sensory information. Classification of the data into several categories of terrain is accomplished using SVM classifier. In order to avoid having to perform classification in a high-dimensional feature space, dimensionality reduction is incorporated implicitly inside the feature extraction process. This terrain classification method exploits features from proprioceptive sensor data. The details of these approaches are presented in Chapter 4 for the robot foot interaction with the ground and in Chapter 5 for the terrain classification based on the proprioceptive sensor data.

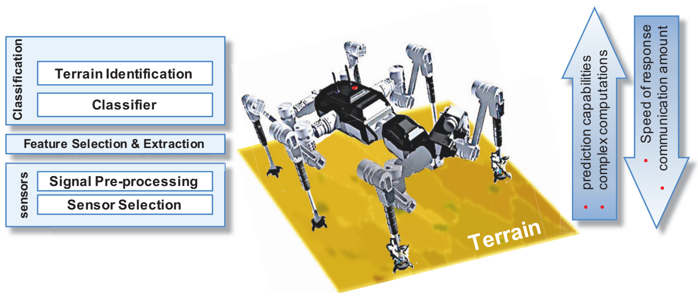


Figure 3.2: Simplified diagram of the main components of terrain identification in our architecture. The four principal elements are: choice of the sensing modalities and sensors (channel selection), preprocessing of the sensor data, feature extraction and classification. The amount of data flow and computation requirements are also shown to the right.

A second terrain classification method is proposed that exploits features from exteroceptive sensor data (image features). This employs a supervised SVM classifier to assign class labels to terrain patches that the rover is about to traversed.

Environment map is built from the surface profile evaluated by the terrain characteristic model and furthermore associated with a semantic interpretation (tile, gravel, grass) based on the terrain type predicted by the visual classifier.

3.2 Terrain Features Acquisition

The data source is generated by the camera and the features computed can be used to classify the terrain type as presented in Chapter 6. Next this will be used to generate samples associating local and remote features. The creation of samples associating local and remote features is based on a grid-based decomposition of the environment, the so-called navigation grid. This grid is formed of cells which are used for the navigation part as a map and a database to store the associations.

3.2.1 Remote Terrain Features Acquisition

At the starting position, the robot builds a map that divides the environment into several numbered areas. These correspond to the navigation cells. As depicted in Figure 3.3, the acquisition of an image at the starting position provides a lot of information. The terrain type of the observed cells can be computed and stored. This is what is done for c_2,c_3, and c_4 .

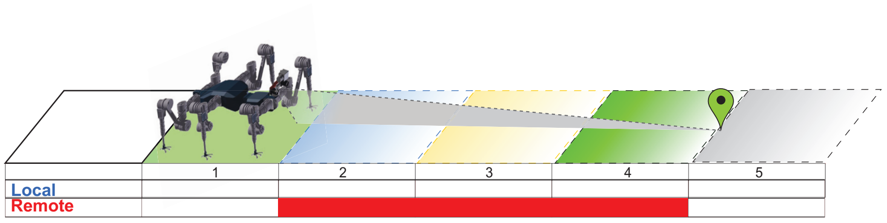


Figure 3.3: Remote terrain features acquisition at start position.

3.2.2 Local Terrain Features Acquisition

Then the robot moves and we can turn our attention to the local robot-foot-ground contact. The data, generated form the data source, is buffered for each one of the footprint contact-points located in a given grid cell. When the contact-point location switches to a different cell, the buffer of the former cell is closed, processed and the corresponding local features are computed.

These information, consisting of the contact force (and other interesting features the designer sees relevant as roughness and slippage) and the cell it belongs to, are saved in a database. Figure 3.4 depicts this acquisition for c_1.

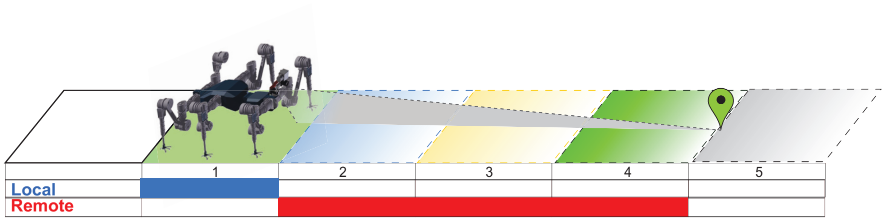


Figure 3.4: Local terrain features acquisition. The robot process its sensors to determine the local terrain features. From which, the robot estimates its interaction with it.

3.2.3 Local and Remote Terrain Features Association

As the robot keeps moving, it finally reaches its goal. Images are frequently acquired and processed while the robot is moving. As a result, remote features are computed for most of the cells the robot traversed. Similarly, the local features are generated and when the position of the cells is behind the robot, and contains both local and far range information, the association between the terrain type information is created and saved for the later use and it could be projected on similarly predicted terrain types. This is illustrated in Figure 3.5 for cells c_2, c_3 and c_4 that now have associations for both local and remote features.

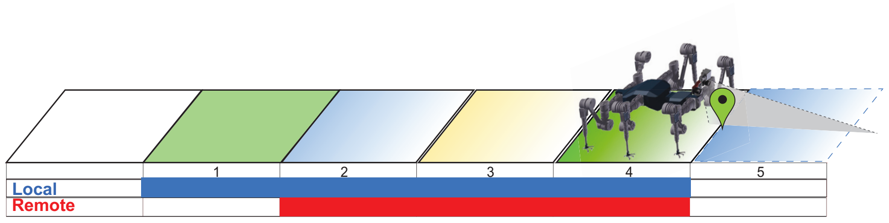


Figure 3.5: Local and remote terrain feature association. The robot associates the local and remote terrain interaction parameters after traversing the environment cells.

3.3 The Experimental Robotic Platform

This section describes the robotic platform used for the development, implementation and testing of the subsystems presented within the scope of this thesis. Basically, the legged robot used in the experiments was the CREX robot designed at DFKI, shown in Figure 3.6.

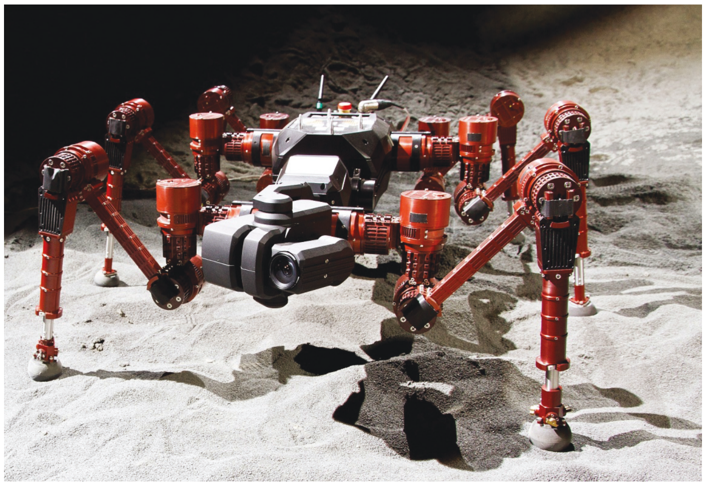


Figure 3.6: CREX the legged robot used in the context of this research.

CREX (CRater EXplorer) is a six-legged walking and climbing robot that is based on the developments in the project SpaceClimber. CREX acts as a scouting system in the project RIMRES. The robot is intended to explore the inner of deep lunar polar craters in search for water ice or other volatile substances.

The key element of the legged robot is the actuator design based on RoboDrive motors. The developed actuator provides a high ratio of output torque to weight (28 Nm continuous torque to approx. 500 g overall weight of the actuator module). The actuator module provides integrated electronics containing power electronics, electronics for sensor data acquisition as well as an FPGA for implementation of control algorithms, logging capabilities and communication with other actuators and the central processing unit. The technical details of the robot are given in Table 3.1.

The robot is equipped with a large number of sensor modalities (see Figure 3.8). We categorize these sensors as follows:

1. Inertial Sensors : This system, developed originally at the University of Michigan, is also know and referred to as the FLEXnav Proprioceptive Position Estimation (PPE) system (Ojeda, Reina & Borenstein, 2004). The FLEXnav PPE system used in this project comprises:
    - Two medium-quality Coriolis gyros for the x and y axis;

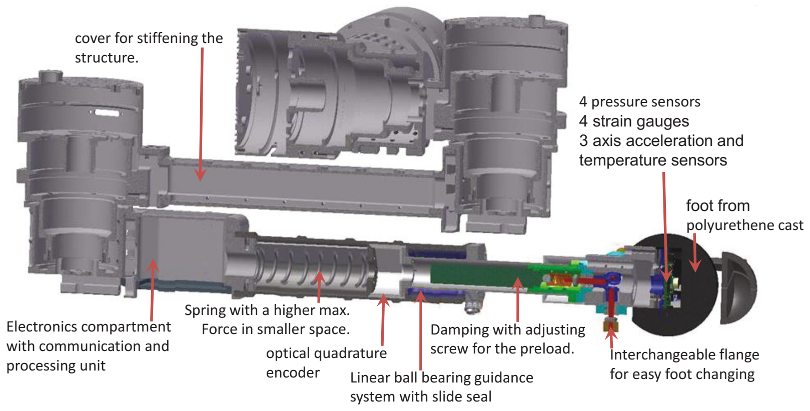

cover for stiffening the structure.
4 pressure sensors
4 strain gauges
3 axis acceleration and temperature sensors
foot from polyurethene cast
Electronics compartment with communication and processing unit
Spring with a higher max. Force in smaller space.
optical quadrature encoder
Linear ball bearing guidance system with slide seal
Damping with adjusting screw for the preload.
Interchangeable flange for easy foot changing

Figure 3.7: Description of the CREX leg and foot.

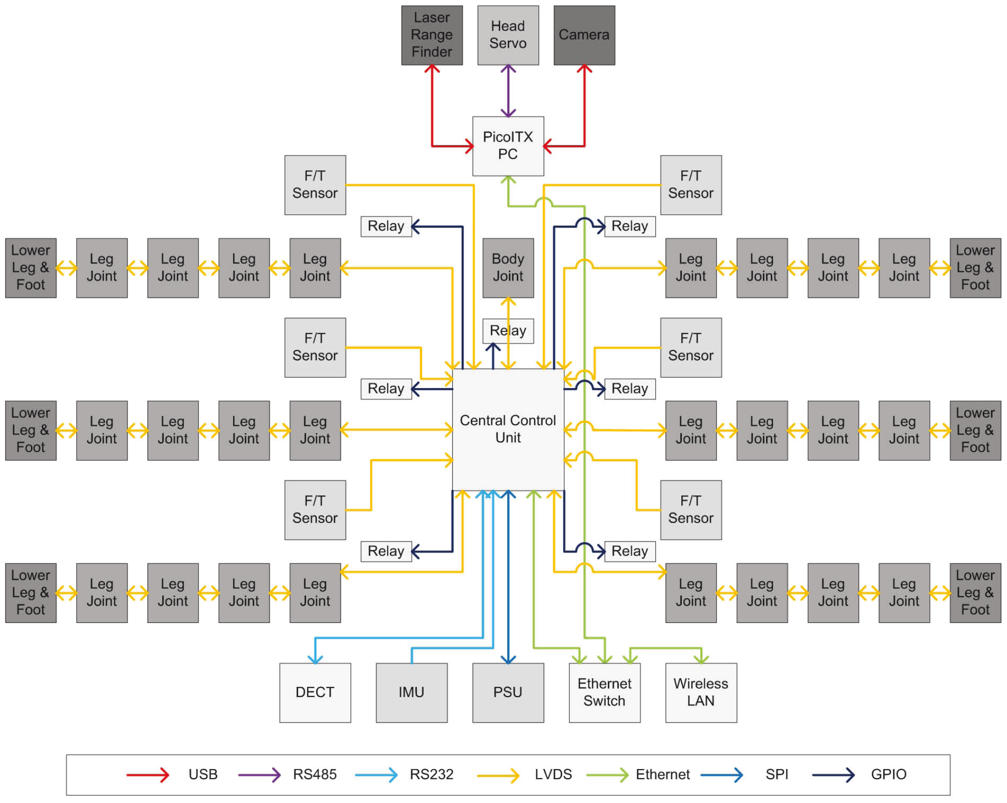


Figure 3.8: CREX inter–module connections for communication

    - One high-quality gyro for the z axis; and
    - Three single-axis accelerometers.
2. Motor Sensors: Two current and two voltage sensors for measuring torque and momentary power consumption for the left-hand and right-hand side motors. While the P2AT has a pair of drive motors on each side, each pair is mechanically coupled and acts effectively as one motor.
3. Range Sensors:
    - One ultrasonic range sensor, which uses a 23 kHz ultrasonic transmitter and one receiver; and
    - One infrared range sensor.

   Figure 1. This Pioneer 2-AT (P2AT) was used in all experiments.
4. Other Sensors:
    - One microphone; and
    - Two wheel encoders.

A laptop computer provided the computing power, and data acquisition was performed using a 16-input 16-bit PCMCIAdata acquisition card. Figure 4 shows a hardware diagram of the complete system, comprising the P2AT platform, computers, and sensors. Software developed specifically for this project includes a library of motion primitives that perform a number of predefined motions. These motions are described in more detail in the following sections.

3.3.1 Locomotion

This part describes the work on MONSTER[1] micro kernel, which is the basis of locomotion control of CREX. The most parts are directly imported from the existing SpaceClimber system. For the changes in hardware, the hardware linking was completely rewritten. Furthermore, the new special behaviors for CREX were also added.

Behavior Architecture

The micro kernel MONSTER [Spenneberg et al., 2005] is a behavior based "operating system", which is developed at DFKI for walking robots. Most part of the SpaceClimber locomotion control software implementation in MONSTER could be reused for CREX. This applies in particular to the parts for walking pattern and posture variations of the robot. Figure 3.9 illustrates the basic structure of this MONSTER–based locomotion control.

MONSTER acts as an interface between the high–level framework (the autonomous behavior) on one-hand and the hardware (FPGA/VHDL-bases) on the other hand. Thereby,

[1]Microkernel for scabrous terrain exploring robots

Table 3.1: Technical specifications of the CREX robot

Dimensions	
Body Dimensions(incl. head)	[L×W×H]: 850×210×180 mm^3
Body Dimensions(stowing pose)	[L×W×H]: 1056×554×227 mm^3
Dimension (normal posture)	[L×W×H]: 820×1000×220 mm^3
Ground clearance	130 mm
Maximum leg length	643 mm
Mass (including batteries)	27 kg
Single leg mass	≈2.6 kg
Body mass (incl. Head)	≈9.4 kg
Motoric System	
Active degrees of freedom	27 DoF (legs: 6×4, body: 1, head: 2)
Passive degrees of freedom	6 DoF (lower legs)
Sensory System	
Proprioceptive sensors	26×motors: position, speed, current, temperature 6× feet: acceleration (three axes), temperature 6× lower leg: spring cylinder immersion 6× leg mounting: six axes force torque 3× body orientation (three axes), supply voltage, overall current
Exteroceptive sensors	6× feet: pressure (4 sensors), 6× feet: (4 strain gauges) head: Laser Range Finder, CMOS camera
Features	
Electro-mechanical interface (EMI)	for tight connection to Sherpa in the project RIMRES. The interface is controlling energy and data exchange between coupled systems
Communication System	
Telemetry and control commands	WLAN (IEEE 802.11g, max.54 Mbit/s, 2.4 Ghz)
Emergency switch	DECT (115.2 kBaud, 1.88 Ghz)
Power Consumption	
Power Supply	50V/2.4mAh Rechargeable battery (Lithium Polymer)
Idle (legs switched off)	≈ 25.5 W
Laid down (legs switched on)	≈ 66.5 W
Standing (in 0°)	≈ 77 W
Walking	100 W – 210 W (depending on walking pattern)

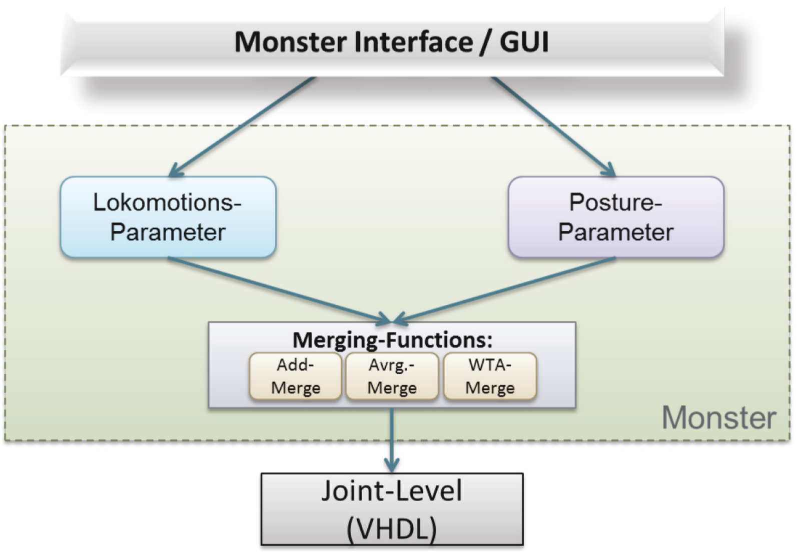


Figure 3.9: Simplified representation of locomotion control of CREX.

MONSTER practically encapsulates the generation of coordinated control of each single joint and leg so only high-level direction commands are needed. These commands can be issued over the proposed architecture to the control framework or over a designeated GUI (see Figure 3.10) which is usually need for for test and demonstration purposes.

On the lowest layer, MONSTER concludes the process with the merging function. In this function, all control information are merged according to spicific merge options and sent to the hardware. This occurs over the LVDS-protocol and the LVDS channels of the hardware boards.

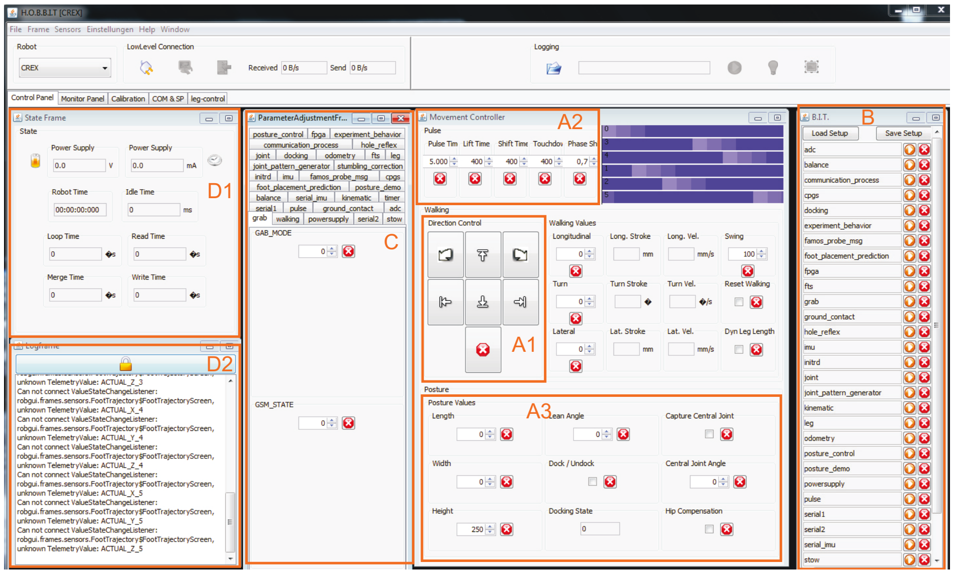
Manual control

In the context of this work the locomotion control of the robot was done through the provided MONSTER. The automatic control of the robot walking was accomplished by issuing direction commands (with specific parameter set) to robot control framework. To design and test these walking commands, the HOBBIT GUI was used. HOBBIT[2] was further developed by the project team to allowed a simplified manual control of the robot for the designed behaviors and other processes. In Figure 3.10, the main window of HOBBIT for manual control is illustrated. The marked parts are for the following functionalists:

A1 basic motion commands: Forward/Backward, Lateral (left/right) and Turning (left/right).

A2 fundamental settings of walking pattern: Cycle time, lift-, shift-, touchdown times.

[2]The same GUI was adapted and used for the same purpose for SpaceClimber



```
unknown TelemetryValue: ACTUAL_Z_3
Can not connect ValueStateChangeListener:
robgui.frames.sensors.FootTrajectory$FootTrajectoryScreen,
unknown TelemetryValue: ACTUAL_X_4
Can not connect ValueStateChangeListener:
robgui.frames.sensors.FootTrajectory$FootTrajectoryScreen,
unknown TelemetryValue: ACTUAL_Y_4
Can not connect ValueStateChangeListener:
robgui.frames.sensors.FootTrajectory$FootTrajectoryScreen,
unknown TelemetryValue: ACTUAL_Z_4
Can not connect ValueStateChangeListener:
robgui.frames.sensors.FootTrajectory$FootTrajectoryScreen,
unknown TelemetryValue: ACTUAL_X_5
Can not connect ValueStateChangeListener:
robgui.frames.sensors.FootTrajectory$FootTrajectoryScreen,
unknown TelemetryValue: ACTUAL_Y_5
Can not connect ValueStateChangeListener:
robgui.frames.sensors.FootTrajectory$FootTrajectoryScreen,
unknown TelemetryValue: ACTUAL_Z_5
```

Figure 3.10: Main window for manual operation of CREX. This GUI is used for the test of behaviors, before this is controlled over the framework.

A3 fundamental options for the modification of posture (body height, left/right- and forward/revers shifting of the body).

B starting and stopping of individual behavior.

C parameter manipulation of walking behavior.

D1 telemetric information: execution time, battery voltage and total running time of system since last start.

D2 warning messages and debug outputs.

"Be sure you put your feet in the right place, then stand firm."

— Abraham Lincoln, (American 16th President, 1809–1865)

4 Robot Foot Ground Contact

In this chapter, a foot–ground contact model is derived. This model uses a point contact dynamics between a robot foot and the ground. In this model, the normal force and shear force to be defined dynamically during the touch down of the foot to the ground. The model presented in the current work is based on traditional analytical wheel–soil modeling approaches of Bekker [Bekker, 1960] and Wong [Wong, 2014].

The analysis and modeling of the foot ground contact for walking robots is currently still a relatively open field. In the area of the wheeled systems there already exist many contributions which deal with its Terramechanics characteristics including analytical [4, 16,23] and finite element [5, 15, 19] methods. Some of these contributions are presented in the next section.

4.1 Traditional Analytical Model Based on Bekker Theory

The Bekker based approach is a traditional wheel–soil model that calculates the stresses and forces acting on the wheel by assuming that the pressure p under a wheel is an exponential function of the form:

$$p = kz^n \tag{4.1}$$

where z is the sinkage, n is the sinkage exponent, and k is an empirical coefficient.

The normal stress distribution under the moving wheel is shown in Figure 4.1 and is often represented by the following equation as described in [Wong, 2014]:

$$\sigma(\theta) = \begin{cases} r^n k(\cos\theta - \cos\theta_f)^n & \text{if } \theta_m \leq \theta < \theta_f \\ r^n k(\cos[\theta_f - \dfrac{\theta - \theta_r}{\theta_m - \theta_r}(\theta_f - \theta_m)] - \cos\theta_f)^n & \text{if } \theta_r \leq \theta < \theta_m \end{cases} \tag{4.2}$$

where θ_f and θ_r are determined by the geometry in Figure 4.1 as follows:

$$\theta_f = cos^{-1}(1 - z/r) \tag{4.3}$$

$$\theta_r = -cos^{-1}(1 - \eta z/r) \tag{4.4}$$

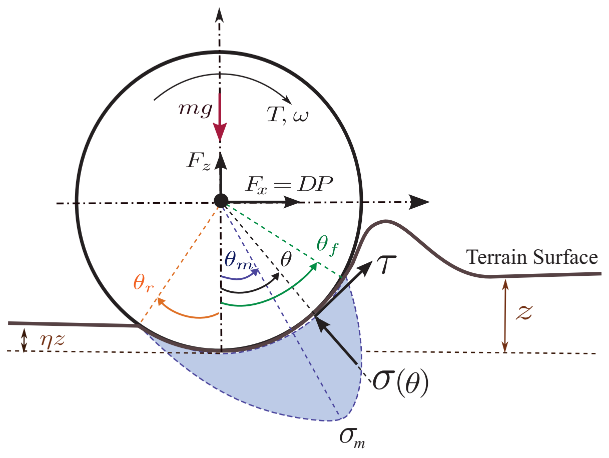


Figure 4.1: Bekker model of stress and forces acting on a wheel. Note: θ_r is measured from the vertical axis and has a negative numerical value in this configuration.

and η is a parameter that is related to the height of the terrain in the track left by the wheel. The angle θ_m at which the maximum normal stress occurs. This angle can be calculated as [Wong, 2014]:

$$\theta_m = (b_0 + b_1 s)\theta_f \tag{4.5}$$

where $b_0 \simeq 0.4$ and $0 \leq b_1 \leq 0.3$. The wheel slip ratio s is defined as:

$$s = \frac{\omega r - V_x}{\omega r} \tag{4.6}$$

where ω is the angular velocity of the wheel with radius r and V_x is the translational velocity of the wheel center.

The shear stress acting along the length of the wheel–soil interface is calculated by Janosi and Hanamoto's equation [Tao et al., 2006]:

$$\tau = (c + \sigma(\theta)\tan\phi)\left[1 - e^{-j(\theta)/K}\right] \tag{4.7}$$

where ϕ is the internal angle of friction, K is the shear deformation modulus, and j can be written in the form used by [Ishigami et al., 2007]:

$$j(\theta) = r[\theta_f - \theta - (1-s)(\sin\theta_f - \sin\theta)] \tag{4.8}$$

The vertical force F_z , drawbar pull (DP), and torque T are area integrals of the stresses acting on the wheel–soil surface as follows:

$$F_z = rb \int_{\theta_r}^{\theta_f} [\tau(\theta)\sin\theta + \sigma(\theta)\cos\theta] d\theta \tag{4.9}$$

$$DP = F_x = rb \int_{\theta_r}^{\theta_f} [\tau(\theta)\cos\theta + \sigma(\theta)\sin\theta] d\theta \tag{4.10}$$

$$T = r^2 b \int_{\theta_r}^{\theta_f} \tau(\theta) d\theta \tag{4.11}$$

Variations and alternative formulations of the pressure–sinkage relationship have been presented in literature. A common pressure–sinkage formulation is given by [Reece, 1965] as follows:

$$p = (ck_c' + \gamma b k_\phi')\left(\frac{z}{b}\right)^n \tag{4.12}$$

where γ is the soil density and c is its cohesion parameters. In this formulation, k_c' and k_ϕ' are two dimensionless pressure–sinkage parameters. From this formulation, it is implied that the pressure p is a function of the terrain's standard material properties: density and cohesion.

Based on Reece's original work, [Wong, 2014] suggests that k_c' is negligibly small; therefore, k_c' is usually set to zero and k_ϕ' is set as a manual tuning parameter for the foot–soil model proposed in this work.

Another common pressure–sinkage relationship is obtained when the solution for k in Equation 4.1 is given by [Bekker, 1960]:

$$k = \frac{k_c}{b} + k_\phi \tag{4.13}$$

substituting this into Equation 4.1 gives:

$$p = (\frac{k_c}{b} + k_\phi) z^n \tag{4.14}$$

where b is the smaller dimension of the contact patch.

It is noteworthy here to state that Equation 4.1 through Equation 4.14 are empirical equations. Moreover, the parameters k_c and k_ϕ are soil constants and their values depend on the experimental conditions used to obtain them.

4.2 Robot Foot Ground Contact Model Based on the Terramechanics Approach

As presented in Section 4.1, the Bekker theory is established for the modeling of wheel-ground contacts (or chain-ground contacts) in the industry and in research. As soil dynamics for the robot has proven to be very important factor to contact treatment, a contact model for the legged robot would be very useful. This contact model can be derived starting on the basis of Bekker theory. For this, however, the existing implementations for wheeled and tracked vehicles operated must be adapted to the characteristics of walking robots.

To derive the foot-ground contact model, we utilize a point contact dynamics between a spherical robot foot and the ground. If the robot leg with its spherical foot touches the homogeneous soil with a certain velocity, the ground produces a normal force $\overrightarrow{F_n}$ and a shear force $\overrightarrow{F_t}$ as shown in Figure 4.2. These contact forces are dynamically changing according to the contact velocity and its direction.

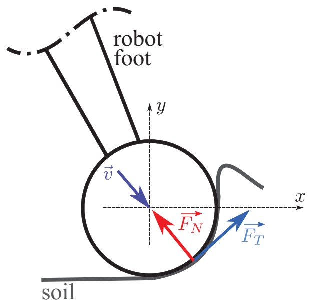


Figure 4.2: Normal (F_N) and shear (F_T) foot–ground contact forces when the leg is moving with velocity $\overrightarrow{v}$

The dynamic normal forces along the abscissa and the ordinate can be modeled by modifying the penalty-based method according to the soil deformation properties as follows:

$$F_{yN} = \begin{cases} k_y dy + p_y \dot{d}y & (dy > 0 \text{ and } \dot{d}y \geq 0) \\ 0 & (dy \leq 0 \text{ or } \dot{d}y < 0) \end{cases} \tag{4.15}$$

$$F_{xN} = \begin{cases} k_x dx + p_x \dot{d}x & (dy > 0) \\ 0 & (dy \leq 0) \end{cases} \tag{4.16}$$

where F_{yN} is the normal force along the Y-axis; F_{xN} is the normal force along the X-axis; k_x and k_y are the shear deformation modules of the soil; p_x and p_y are the pressure components on the soil.

The plastic equilibrium model for soils is commonly used [Sancho-pradel and Gao, 2010] to analyze the forces and stresses produced by the interaction between the robot chassis (e.g. wheels, legs, etc.) and the soil. In this model, the Mohr–Coulomb's shear failure criterion can be used for relating the shear strength of the soil τ_{max} to the corresponding compressive normal stress σ at the contact area through the extending the Bekker model as follows:

$$\tau_{max} = c + \sigma \tan\phi \tag{4.17}$$

where c is the soil cohesion and ϕ is the internal friction angle.

These forces have scientific and engineering relevance. As a robot capable of performing in situ estimation of these forces could provide invaluable data for the characterization of the soil.

This relationship is derived from the generally accepted approximation of the shear stress from [Janosi and Hanamoto, 1961] given as:

$$\tau = \tau_{max}\left(1 - e^{-j/K}\right) \tag{4.18}$$

where j is the soil shear deformation given by Equation 4.8 and K is the shear deformation modulus.

Equation 4.18 can be used to calculate the shear stress of the foot–ground contact. The horizontal and vertical shear stress components can be described as follows:

$$\tau_x = \left(c + F_{yN}\, tan\phi\right)\left(1 - e^{-j_x/K}\right) \tag{4.19}$$

$$\tau_y = (c + F_{xN}\, tan\phi)\left(1 - e^{-j_y/K}\right) \tag{4.20}$$

Therefore, these shear stress components (Equation 4.19 and Equation 4.20) can be employed into the normal shear of the foot–ground contact from the penalty-based model approach. The advantage of this contact model is that the normal and shear stresses are dynamically defined by the foot-to-surface contact.

4.3 Plastic Terramechanics Particle (PTP) Model

In many previous researches of planetary exploration and extraterrestrial systems, analytical approaches such as Bekker's mobility model [Bekker, 1960] and Terzaghi's soil deformation model [Terzaghi, 1943] have been widely used to simulate soil contact behaviors of wheel vehicle systems and excavation systems. However, the main drawbacks of such analytical models are that:

- a lot of soil parameters such as angle of internal friction, soil stiffness, soil cohesion, shear deformation modulus, soil density, etc., are necessary to describe soil contact models,

- the accuracy of measured soil parameters cannot be guaranteed in all test environments because the parameters can change depending on the states of temperature or humidity,
- mathematical formulas of contact models strongly depend on the body-shape of contact objects, and
- it is difficult to describe a contact model for a heterogeneous soil because of the complex soil parameters.

The Bekker's theory is originally developed and used for wheeled systems. In this section this theory is further extended to develop a *particle–based* contact model for the robot foot terrain interaction. This new approach of a realistic soil contact model using a particle system is based on its point contact model concepts. The plastic terramechanics particle (PTP) system is developed to achieve real-time 3-dimensional contact behaviors between rigid body objects and the soil.

Assume the soil surface when contacting a rigid robot foot has plastic behaviors such as in Figure 4.3. When the rigid foot penetrates into the soil, the soil will give a reaction to the rigid foot with a deformation of the soil, whereas the soil shape deformed by the contact is kept when the rigid foot pulls back.

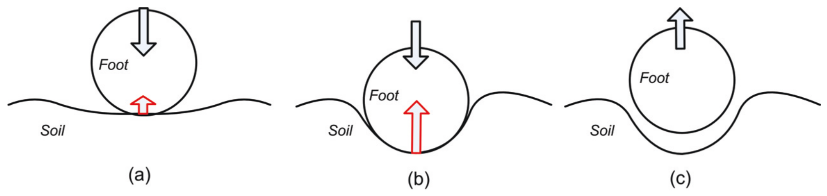


Figure 4.3: The plastic behavior of the soil surface during contacts

In our model, a *plastic terramechanics particle* (PTP) is proposed to analyze and calculate the 3D contact behaviors between robot feet and soil surface. A PTP soil particle is shown in Figure 4.4.

The properties of the PTP model are as follows:

- Soil is splitted into one or more layers of volume cells. Each cell contains a single plastic terramechanics particle (PTP) at its spatial center (Figure 4.5-(a)).
- A PTP is a physics model which has 3D plastic behavior with contact.
- Each PTP has $x-y-z$ plastic parameters which define its 3D Plastic behavior (represented by the spring damper symbols as shown in Figure 4.4) at its contact point(s) with the object (e.g, robot foot).
- Reaction forces generated by PTPs can be combined together and then applied to the contact object.

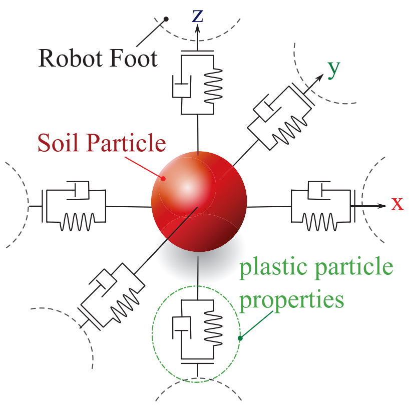


Figure 4.4: The plastic terramechanics particle (PTP) with its force parameters that define the resulting force at the contact point with the robot foot. The parameters are represented with the spring damper symbols.

- In contrast with standard particle systems [Zang and Zhao, 2013], PTPs are independent of each other (i.e., the reaction force from a PTP can not be transferred to another PTP).
- Through contacts with objects, PTPs can move from one cell to another. As illustrated in Figure 4.5, an object and a 2-dimensional surface cross-section that consists of cells A_{00}–A_{34}. Every cell initially contains a single PTP. When the object penetrates the soil perpendicularly (Figure 4.5-(c)), all PTPs in contact with the object will move as shown. Thus, the cells A_{01}, A_{02}, A_{03}, and A_{12} have zero PTPs, whereas the cells A_{11} and A_{13} have two PTPs and A_{22} has three PTPs respectively. A possible diagonal movement in 1D is shown.

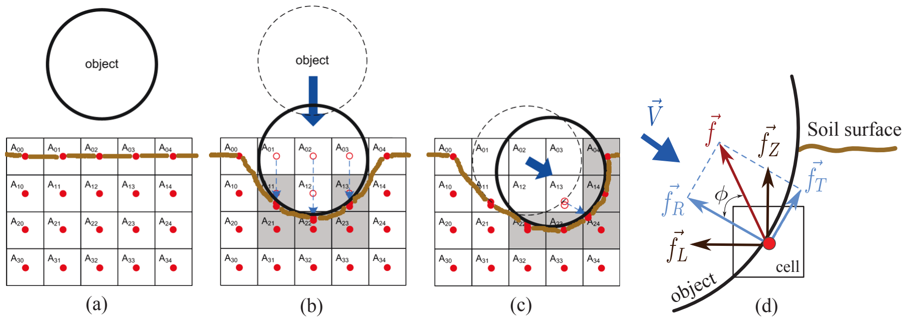


Figure 4.5: PTP–object contact behavior. (a) The soil is discretized in three spatial dimensions (here only two are shown), and each cell is filled with one PTP at its center. (b) and (c) objects are able to move PTPs from one cell to another. If an object is in contact with at least one PTP of a cell, that cell exerts a force on the object. (d) The magnitude of this force depends on the number of particles residing in the cell.

- If at least one PTP in a cell is in contact with the object (e.g., a robot foot) that penetrates the cell, the cell generates a reaction force in the opposite direction of the object movement. The exerted reaction force can be calculated based on of Bekker pressure-sinkage (Equation 4.21) on the rigid body:

$$p = \left(\frac{k_c}{b} + k_\varphi\right) z^n \tag{4.21}$$

where p is the pressure, k_c is the cohesive modulus of soil deformation, k_φ is the frictional modulus of soil deformation, n is the soil-specific empirically-determined exponent, and b is contact area.

- When having a rigid body-soil contact, PTPs exert reaction forces of a shear stress (Equation 4.22) on the rigid body. This force is based on the Janosi and Hananoto relation:

$$\tau = (AC_0 + \sigma \cdot tan\phi)\left\{1 - e^{(-j/K)}\right\} \tag{4.22}$$

where j is the shear deformation, K is the shear deformation modulus, τ is the shear stress of the soil for a given normal stress σ, C_0 is the cohesion stress of the soil, A is the contact area, ϕ is the angle of internal shearing resistance of the soil.

- Also, when having a rigid body-soil contact, PTPs exert reaction forces of the bulldozing–horizontal force as in Figure 4.6-(a) and excavation force as in Figure 4.6-(b) on the rigid body.

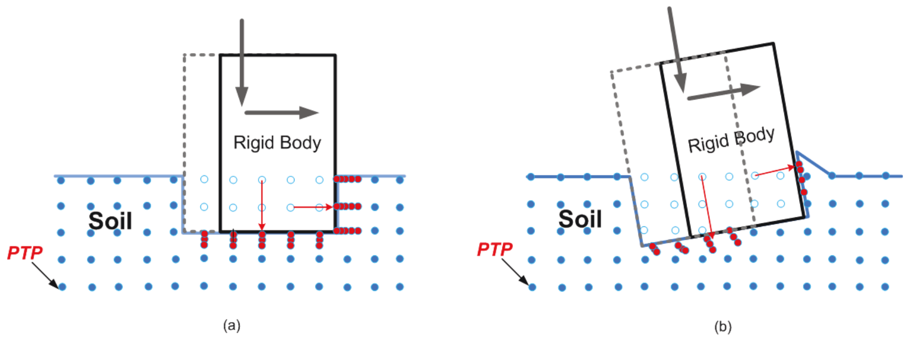


Figure 4.6: Bulldozing force with the plastic terramechanics particles (PTPs): (a) in horizontal behavior and (b) in excavation behavior

- The total reaction force f from the cell is described as:

$$\mathbf{f} = \mathbf{f_Z} + \mathbf{f_L} = \mathbf{f_R} + \mathbf{f_T} \tag{4.23}$$

where $\mathbf{f_Z}$ is the vertical force, $\mathbf{f_L}$ is lateral force, $\mathbf{f_R}$ is the force perpendicular to the object contact surface, and $\mathbf{f_T}$ is the force tangential to the object contact surface (see Figure 4.5-(d)).

Equation 4.23 can be rewritten as:

$$F_f(F_v, j) = [A \cdot C_0 + F_z \cdot \tan(\phi)]\left(1 - e^{-j/K}\right) \tag{4.24}$$

where A is the surface area of contact and F_z is the vertical force. C_0, ϕ and K are the cohesion, the angle of internal friction and the modulus of the shear deformation of the soil, respectively. In this relation, j is the lateral displacement of the particle from its original position. On the other hand, the cell particles exert a force F_b as:

$$F_b(j) = (V \cdot B)\left(1 - e^{-j/K}\right) \tag{4.25}$$

where V is the volume that is represented by a particle and B is the *Bulldozing parameter* that can be determined through experiments. j and K are defined the same as in (Equation 4.24. Through this relation, the Bulldozing effect on the robot foot can be modeled.

This yields a resultant lateral force:

$$F_l(F_v, j) = [A \cdot C_0 + F_z \cdot \tan(\phi) + V \cdot B]\left(1 - e^{-j/K}\right) \tag{4.26}$$

It should be noted that the particle exerts both F_Z and F_L on the robot foot. This sum of the particle forces will slow down the robot foot penetrating the soil and brings it to a halt at the end.

- The number of PTPs in a cell represents the soil property of pressure P through a look up table which is found by simple pressure-sinkage experiments.

4.3.1 Discretized PTP Model

For a real-time capable implementation of of this approach, the theoretical pressure-sinkage relation (Equation 4.1) can be modified as proposed by Goriatchkin to incorporate the effect of particle density [Meirion-Griffith and Spenko, 2014]. As can be obviously inferred, the density of the particles has an influence on the physical properties of the soil in the model. In the simple case of the vertical force with exponent $n = 1$ follows the modeled pressure:

$$p(z) = k \cdot z^n \quad \xrightarrow{n=1} \quad p(z) = \frac{k}{2} \cdot \left(\frac{z^2}{\Delta z} - z\right) \tag{4.27}$$

where Δz is the vertical distance between two particles layers. This relationship is illustrated in Figure 4.7.

These results are used for preliminary check of the proposed PTP model accuracy. From such data the pressure lookup table is created by sampling the data according to the soil model cell density.

It is well noticed that this approach has strong dependency on the exact analytical description of the contact forces and the need of precise values of soil parameters. The presented PTP model needs further modification to overcome these limitations.

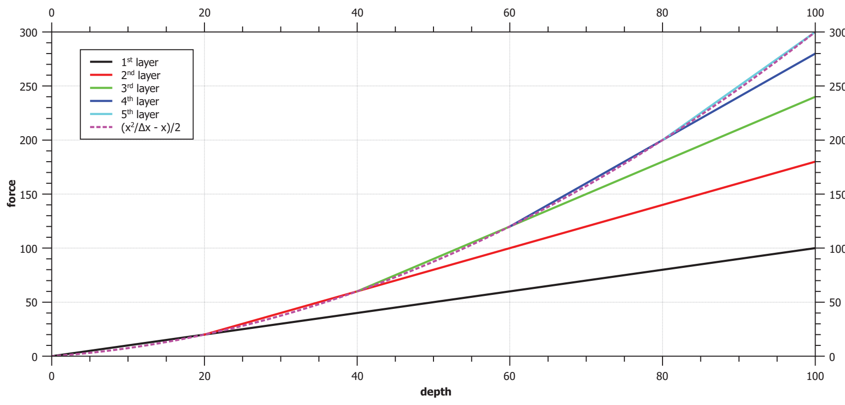


Figure 4.7: Impact of several layers of particles in the PTP model. Here, the vertical force for $n = k = 1$ and $\Delta z = 20$ is shown.

To avoid the dependency of the modeled soil pressure on layer density, the reaction pressure of the soil is not generated from the particles themselves but from virtual cells. These virtual cells are obtained by the discretization of the soil and the particles are placed at their centers (see Figure 4.5).

The magnitude of the pressure from a cell, in contact with an object, is determined through the number of particles located in the cell. Therefore, not only the soil is discretized but also its reaction force. Thus, it is possible to drop the analytical model of the terramechanics and store the required values instead in a lookup table.

These lookup table values of the pressure as a function of penetration depth can be determined through simple experiments. From the experiment results, the correlation between penetration depth and reaction force can be determined. These values are then sampled for a selected particle density. Figure 4.8 shows a typical example of this procedure

To calculate the reaction force of the cell with grid location (i, j, k), a simple lookup function λ using the number of particles in the cell is performed and multiplied by the area of the cell. The total force is simply the sum of all cell forces:

$$\mathbf{f_z} = \sum_{i,j,k} \mathbf{f}_{i,j,k} = A_c \sum_{i,j,k} \lambda(N_{i,j,k}) \cdot \mathbf{e}_z \tag{4.28}$$

where A_c is the cell area, λ is the lookup function, and $N_{i,j,k}$ is the number of particles in the cell at location (i, j, k). The direction of this force is opposite to the direction of the foot movement as shown in Figure 4.5. While Equation 4.28 gives the derivation of the vertical force f_z, the same process can be used to compute the lateral force f_L to obtain the total force in Equation 4.23.

As illustrated in Figure 4.8, a higher resolution of the particles gives a finer discretization of the Pressure-sinkage force and therefore a better approximation to the real interaction

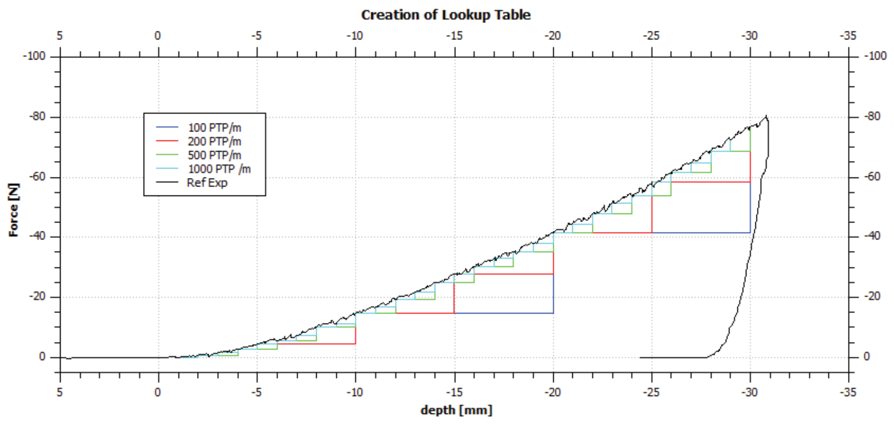


Figure 4.8: Sampling a pressure-sinkage curves for different particle densities.

between robot leg and the soil. Another advantage of this method is that the robot can reuse a previously produced lookup table for a specific terrain type when it is recognized during a mission.

4.4 Summary

The Bekker's theory is originally developed and used for wheeled systems. In this chapter this theory is further extended to develop a *particle–based* contact model for the robot foot terrain interaction based on its point contact model concepts.

This new approach is a realistic soil contact model using a modified particle system. The plastic terramechanics particle (PTP) system is developed to model real-time 3-dimensional contact behaviors between rigid body objects and the soil. For a real-time capable implementation of this approach, its reformulated to use a lookup table generated from simple contact experiments of the robot foot with the terrain.

“Look up at the stars and not down at your feet. Try to make sense of what you see, and wonder about what makes the universe exist. Be curious.”

— Stephen Hawking, (British theoretical physicist, and cosmologist

5 Short–Range Embodied Terrain Classification

5.1 Machine Learning Tools

We performed the classification task using several machine learning tools. Therefore, we used the machine learning software pySPACE described in Section 5.3, in addition to own implementation of some of them to train and test these classifiers. The adopted methods are Support Vector Machine (SVM), Linear SVM, and k-means. All of them are effective tools of machine learning and many are good for novelty detection.

5.2 Support Vector Machines

Support vector machines (SVMs) ([Cortes and Vapnik, 1995, Byun and Lee, 2002]) are supervised learning machine learning models with associated learning algorithms that analyze data to recognize patterns. They take an input vector and determine its class out of a given set of classes.

The principle of the SVM is to construct two parallel hyperplanes with maximum distance such that the samples belonging to different classes are separated by the space between these hyperplanes (see also Figure 5.1). Such space between the planes is usually called margin—or *inner margin*.

The SVM makes a model based on the training data and then classifies further input data into the learned classes. The underlying model consists of points in space representing training examples. These examples are mapped into separate categories with wide boundaries between them. New input data are also mapped and then predicted to belong to a category based on their closeness. SVM actually creates a set of hyperplanes in a high dimensional space. It can also perform non-linear classification using a technique called kernel trick, i.e. mapping their input into high dimensional feature spaces (Chang

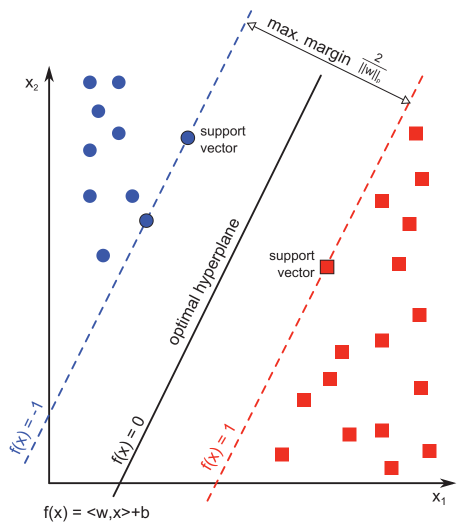


Figure 5.1: Support Vector Machine Scheme. The blue dots are training samples with $y = -1$ and the red dot with $y = +1$ respectively. Displayed are the three parallel hyperplanes H_{+1}, H_0, and H_{-1}.

and Lin, 2005; Erasto, 2001; Hsu et al., 2003). SVMs are extensively used for machine learning in different fields of science (Mittag et al., 2012).

The hyperplane algorithm by the SVM developer (Vapnik and Lerner, 1963) was a linear classifier. Later Vapnik and others (Boser and Vapnik, 1992) proposed to use the *kernel trick* to create a non-linear classifier. The main algorithm remains the same except that every dot product is replaced by a non-linear kernel function (κ). This results in a nonlinear separation of the data which is very advantageous, if the data is not linearly separable. The respective classification function becomes:

$$f(x) = \sum_{i=1}^{n} \alpha_j \kappa(x, x_i) \tag{5.1}$$

This allows for a fitting of the maximum margin hyperplane in a transformed feature space. The most common kernel functions are displayed in Table 5.1.

For some applications like text or graph classification, a kernel function is directly applied to the data without the intermediate step of creating a feature vector. Using a Gaussian radial basis function as a kernel results in the feature space becoming a Hilbert space of infinite dimensions.

Commonly, only the Euclidean norm[1] is used for measuring the distance between points but it is also possible to use an arbitrary p-norm $\left(\|v\|_p = \sqrt[p]{\sum v_i^p}\right)$ with $p \in [1, \infty]$.[2] For

[1] $\|v\|_2 = \sqrt{\sum v_i^2}$
[2] Due to convergence properties it holds $\|v\|_\infty := \max |v_i|$.

Table 5.1: Support vector machine kernel function types. Kernel functions are applied to the input (x_i, x_j). The other variables in the functions are additional hyperparameters to customize/tune the kernel for the respective application.

Kernel (κ)	**Function**
linear kernel	$\langle x_i, x_j \rangle$
polynomial kernel	$(\gamma \langle x_i, x_j \rangle + b)^d$
sigmoid kernel	$\tanh(\gamma \langle x_i, x_j \rangle + b)$
Gaussian kernel (RBF)	$\exp\left(-\frac{\lVert x_i - x_j \rVert_2^2}{2\sigma^2}\right)$
Laplacian kernel	$\exp\left(-\frac{\lVert x_i - x_j \rVert_2}{\sigma}\right)$

getting instead the distance between two parallel hyperplanes or a point and a hyperplane, the respective dual p'-norm has to be used with $\frac{1}{p} + \frac{1}{p'} = 1$ [Mangasarian, 1999]. If $p = \infty$, p' is defined to be 1. Having the two hyperplanes H_{+1} and H_{-1} with:

$$H_z = \{x \mid \langle w, x \rangle + b = z\}, \tag{5.2}$$

their distance is equal to $\frac{2}{\lVert w \rVert_{p'}}$. Hence, the resulting model reads as:

Method 1 (Maximum Margin Separation).

$$\begin{array}{llll} \max\limits_{w,b} & \frac{1}{\lVert w \rVert_{p'}} & & \\ s.t. & y_j(\langle w, x_j \rangle + b) & \geq 1 & \forall j : 1 \leq j \leq n. \end{array} \tag{5.3}$$

In this model, $\mathbf{x} = \{x_1, \cdots, x_n\}$ is the observation input vector with associated class label y_i. The dot product is denoted by $\langle , \rangle$ and $\mathbf{w}$ is the normal vector to the hyperplane. The offset of the hyperplane from the origin along the normal vector $\mathbf{w}$ is $\frac{b}{\lVert w \rVert_{p'}}$.

For better solvability, Model 1 is reformulated to an equivalent one. The Euclidean norm is used ($p = p' = 2$) and the fraction in Model 1 is inverted. Additionally, the root is squared and a scaling factor of $\frac{1}{2}$ is added to get better properties of the derivative which simplifies the optimization process and further calculations. Finally, the respective minimization problem is solved instead. These superficial modifications do not change the optimal solution. The resulting reformulated model reads:

Method 2 (L1–Support Vector Machine).

$$\begin{array}{llll} \min\limits_{w,b,t} & \frac{1}{2} \lVert w \rVert_2^2 + C \sum t_j & & \\ s.t. & y_j(\langle w, x_j \rangle + b) & \geq 1 - t_j & \forall j : 1 \leq j \leq n, \\ & t_j & \geq 0 & \forall j : 1 \leq j \leq n. \end{array} \tag{5.4}$$

In the modified model (2), some disturbance in the form of samples penetrating the margin is allowed. This is denoted with the error variable t_j. This is to account for the fact that strict separation margins are prone to overfitting or do not exist at all.

The hyperparameter C (called regularization constant, cost factor, or complexity) defines the compromise between the width of the margin $\left(\text{regularization term } \frac{1}{2}\|w\|_2^2\right)$ and the amount of samples lying in or on the wrong side of the margin $\left(t_j > 0\right)$. This principle is called *soft margin*, because the margins defined by the two hyperplanes H_{+1} and H_{-1} can be violated by some samples (as shown in Figure 5.2).

In the final solution of the optimization problem only these samples and the samples on the two hyperplanes are relevant and provide the SVM with its name (see Definition 1).

Definition 1 (Support Vectors)**.**
The vectors defining the margin, i.e., those data examples x_j where $t_j > 0$ or where identity holds in the first inequality constraint (Method 2), are the *support vectors*.

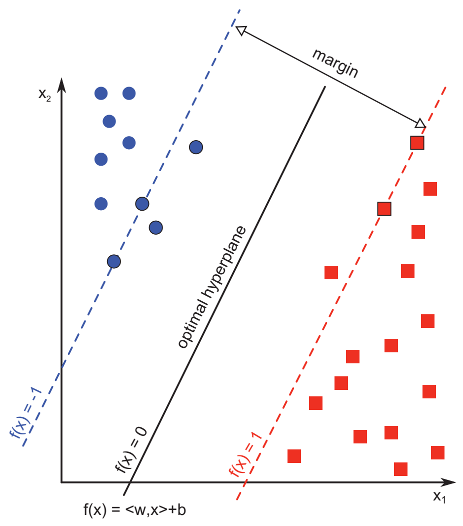


Figure 5.2: Soft Margin Support Vector Machine Scheme. In contrast to Figure 5.1, some samples are on the wrong side of the hyperplanes within the margin.

The L1 in the method name of the SVM (Method 2) refers to the loss term $\|t\|_1 = \sum t_j$ for $t_j \geq 0$ in the target function.

5.2.1 Linear SVM

A linear classifier is a classifier which performs the classification by a linear combination of the features of the dataset. It is a very fast method to solve classification problems of extremely large datasets [Burges, 1998]. It is supposed to be linearly scalable, which means that it uses an SVM model with a running time that scales linearly with increasing size of a typical dataset [Fawzi et al., 2014]. Thus it can solve multi-class classification problems in linear time.

The L1–Support Vector Machine with soft margin given in Method 2 and Equation 5.4 is used. For new data, the respective linear classification function is:

$$f(x) = \left\langle w, x_j \right\rangle + b. \tag{5.5}$$

To get a mapping to the class labels −1 and +1 we use the decision function:

$$y(x) = \operatorname{sgn}(f(x)) := \begin{cases} +1 & \text{if } f(x) > 0, \\ -1 & \text{otherwise.} \end{cases} \tag{5.6}$$

5.2.2 Multiclass SVMs

SVMs are inherently two-class classifiers. The traditional way to do multiclass classification with SVMs is to use one of the methods discussed in [Hsu and Lin, 2002] and [Duan and Keerthi, 2005].

The straight forward scheme to classify among more than two distinct groups is to construct a *one-vs-one* multiple SVM's. This scheme constructs one binary classifier for every pair of distinct classes and so, all together $\binom{n}{2} = \frac{n(n-1)}{2}$ binary classifiers for n classes are constructed. The binary classifier C_{ij} is trained taking the incoming feature vector from c_i as positive and the examples from c_j as negative. For a new example $\mathbf{x}$, if classifier C_{ij} says $\mathbf{x}$ is in class c_i, then the vote for class c_i is added by one. Otherwise, the vote for class c_j is increased by one. After each of the $n(n-1)/2$ binary classifiers makes its vote, the final classification strategy assigns $\mathbf{x}$ to the class that scores the max-wins voting.

In this `one-vs-one` scheme, the time for training classifiers may actually decrease, since the training data set for each classifier is much smaller. A significant drawback of this scheme is its factorial grow with the number of classes. It also suffers from false positives if a feature vector is queried that does not belong to any of the classes [Duan and Keerthi, 2005].

The other most common technique in practice is the *one-versus-all* scheme (also referred to as "one-versus-rest"). This technique constructs n binary classifiers in which the i^{th} classifier is trained to classify a feature vector $\mathbf{x}$ as either belonging to class c_i, or belonging to class $\neg c_i$. For a new example $\mathbf{x}$, this technique assigns it to the class with the largest value of the output function.

Figure 5.3 Schematically illustrates `one-vs-all` concepts. When the query feature vector is of class A, the `A-vs-all` SVM will classify it correctly, and thus the overall output will place it into class A. When the query is of a different class, D, which is not already existent in the class structure, the query will always fall into the `All` class on the individual SVM's. Hence, the query will not be falsely categorized into any class.

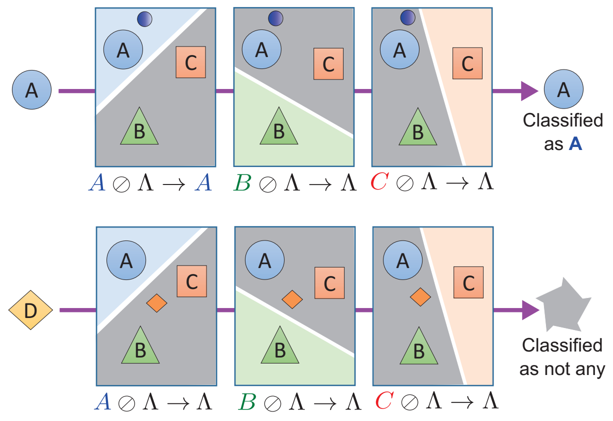

A
A
C
B
$A \oslash \Lambda \to A$
$B \oslash \Lambda \to \Lambda$
$C \oslash \Lambda \to \Lambda$
Classified as A
D
$A \oslash \Lambda \to \Lambda$
$B \oslash \Lambda \to \Lambda$
$C \oslash \Lambda \to \Lambda$
Classified as not any

Figure 5.3: Schematic of `one-vs-all` SVM. When query sample is of class A, the `A-vs-all` SVM will correctly classify it. When it is not of class A, B, or C, it will likely not be classified into any. Here, $\oslash$ and Λ stand for vs. and `All`, respectively.

One advantage of this approach is its interpretability. Since each class is represented by one and one classifier only, it is possible to gain knowledge about the class by inspecting its corresponding classifier. In addition, the number of SVM's needed in a `A-vs-all` scheme only grows linearly with the number of classes, n. This system is also more robust scheme as it does not suffer from as many false positives because samples that do not belong to any of the classes are usually classified as such in each individual SVM.

5.3 pySPACE Tool

For the different machine-learning tasks (e.g., finding the best parameters of the classifier and the minimum number of sensors needed for optimum performance), we used the Python Signal Processing And Classification Environment (pySPACE) [pySPACE, 2015].

pySPACE incorporates a large number of machine learning algorithms. It has algorithms in many categories of machine learning techniques. The software provides automated data handling, distributed processing, modular build-up of signal processing chains and tools for visualization and performance evaluation. Included in the software are various algorithms like temporal and spatial filters, feature generation and selection, classification algorithms, and evaluation schemes. Further, interfaces to other signal processing tools are provided. pySPACE originally has been built to process multi-sensor windowed time series data [Krell et al., 2013]. Figure 5.4 shows an overview of pySPACE algorithms and node types.

The software has a mechanism to compare algorithms applied automatically on time series dataset, either with the aim of finding a suitable preprocessing, or of training supervised algorithms to classify the data. This can be done by splitting the dataset into

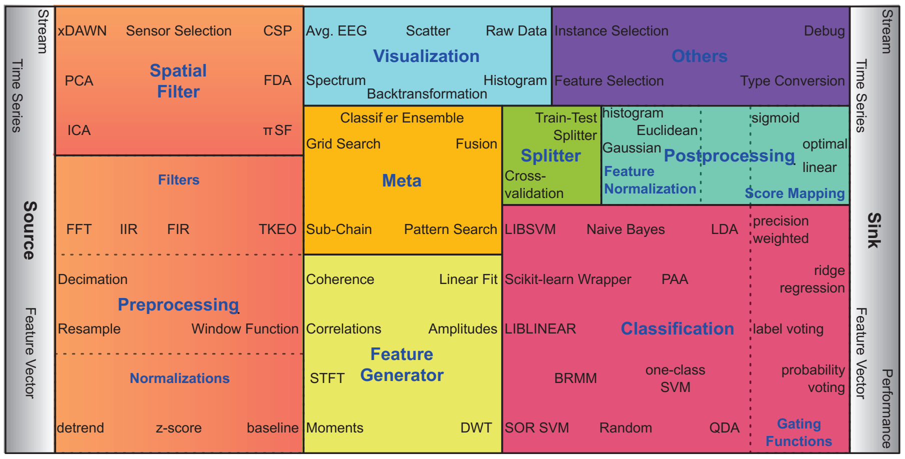


Figure 5.4: Overview of pySPACE algorithms and node types. The software currently includes more than 100 processing nodes arranged in figure according to processing categories (package names) and subcategories [Krell et al., 2013].

training and test subsets, or by using cross-validation. In the following sections, the relevant principles of pySPACE processing are presented.

5.3.1 Data

The data to be used within pySPACE are distinguished by its granularity: from single data samples to datasets and complete summaries. Four types of *data samples* can occur in pySPACE: raw data stream, windowed time series, feature vector, and prediction vector.

These different data types require at the same time different types of processing including ways to load data into the framework and how the outcome is stored. The data organization is illustrated in Figure 5.5.

To load a data sample, it must come with a metadata file (in YAML) that gives additional description of the data such as specifying sensor names, sampling frequency, feature names, or classifier information. When loading a *raw data stream* it is first of all segmented into a *windowed time series*. Windowed time series have the form of two-dimensional arrays with amplitudes sorted according to sensors on the one axis and time points on the other. *Feature vectors* are one-dimensional arrays of feature values. In a *prediction vector* the data sample is reduced to the classification outcome and the assigned label or regression value.

For analysis, data samples are combined to *datasets* (defined as a recording of one single experimental run), either as streamed data or already preprocessed as a set of the corresponding time series windows, or as a loose collection of feature vectors. It also has metadata specifying the type, the storage format, and information about the original data and preceding processing steps.

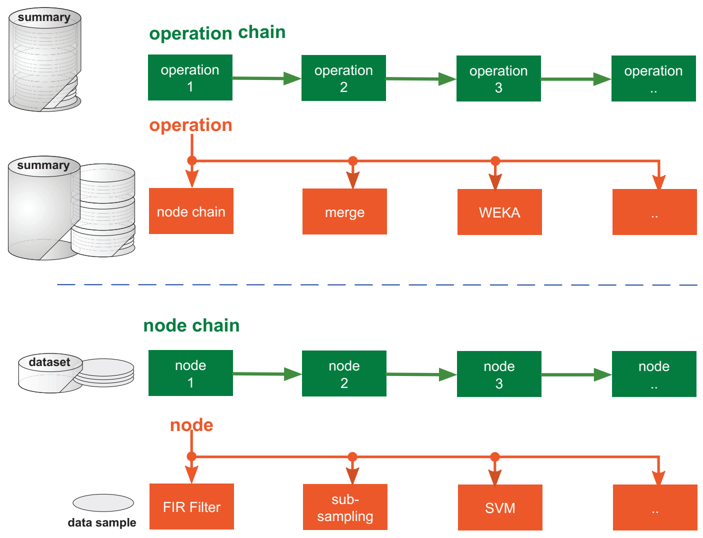


Figure 5.5: Data types (summary, dataset, sample) and processing operations (operations, nodes, and algorithms) in pySPACE. For the operations and nodes several different algorithms can be chosen. Algorithms are depicted in orange and respective infrastructure components concatenating these in green. Adopted from [Krell et al., 2013].

5.3.2 Algorithms: Nodes and Operations

Nodes and operations are the low and high -level algorithms in pySPACE (see Figure 5.4). Nodes are the signal processing algorithms that operate on data samples (e.g., single feature vectors). Some nodes are trainable, i.e., they define their output based on the training data provided. They are grouped depending on their functionality as depicted in Figure 5.4.

An operation automatically processes one data summary and creates a new one. It is also responsible for the mapping between summaries and datasets. Operations can be used, among many others, to reorganize data (e.g., shuffling or merging) and visualize results. The node chain operation is the most important operation that enables automatic processing of a sequence of nodes.

a node chain is a concatenation of nodes to perform a desired signal processing flow. The input of a node chain is a dataset which is accessed by a *source* node at the beginning of the node chain. A sink node is at the end of the node chain to collect the result and return a dataset as output.

5.4 Sensor Data Processing

pySPACE software is used to analyze and compare the performance (e.g., classification accuracy) of different methods and parameter settings, respectively. It allows to estimate these quantities based on different validation schemes (including cross-validation) and several independent runs.

The experiment data are stored in the comma separated values (.csv) format which is supported by pySPACE (the machine learning framework).

The specification of the log files containing the data to be processed are prepared in a text file with a specific format (YAML). This data file is then loaded by pySPACE into a database and various machine learning operations are performed.

Groups of datasets, e.g., experimental repetitions with the same terrain or different terrains, are combined to be analyzed and compared jointly. Such dataset collections are called *summary* in pySPACE. Summaries are organized in folder structures. To enable simple evaluations, all single performance results in a summary are combined to one csv file, which contains various metrics, observed parameters and classifier information.

For a complete processing of data from time series windows over feature vectors to the final predictions and their evaluation, several processing steps are needed as outlined in Figure 5.6.

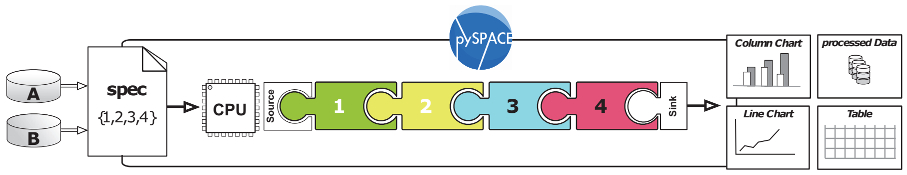


Figure 5.6: Processing scheme of a node chain operation in pySPACE. A and B are two different experiment datasets, which shall be processed as specified in the spec file. The processing is then performed automatically. As a result, it produces the new processed data data and also visualizations and performance charts. The puzzle symbols illustrate different modular nodes, e.g., a cross-validation splitter (1), a feature generator (2), a visualization node (3), and two a classifier (4). They are concatenated in a node chain.

The major steps performed on the data are given as follows: *Preprocessing* comprises denoising time series data and reducing dimensionality in the temporal with spatial filters. This can be done by combining the signals of different sensors to new virtual sensors or by applying sensor selection mechanisms. Here we employed the sensor selection mechanism to find the efficient minimum set of sensors that gives the optimal performance.

Classification algorithms typically operate on feature vector data, i.e, before classification the time series have to be transformed with at least one *feature generator* to a feature vector. A classifier is then transforming feature vectors to predictions. In *postprocessing*, feature vectors can be normalized and score mappings can be applied to prediction scores. For every data type a *visualization* is possible. Furthermore, there are *meta* nodes, which internally call other nodes or node chains. Thus, they can combine results of nodes or optimize node parameters. If training and testing data are not predefined, the data must be *split* to enable supervised learning. By default, data are processed as testing data.

Source nodes are necessary to request data samples from the datasets, *sink* nodes are required for gathering data together to get new datasets or to evaluate classification performance. They establish the connection from datasets to data samples which is required for processing datasets with concatenations of nodes.

"I think setting a goal, getting a visual image of what it is you want. You've got to see what it is you want to achieve before you can pursue it."
— Chuck Norris, (American martial artist, actor, film producer and screenwriter)

6 Long–Range Visual Terrain Classification

This chapter presents our visual terrain classification system. Figure 6.1 illustrates its general overview. It uses a single camera to acquire terrain images. The acquired images are classified by a feature-based terrain classification algorithm in a supervised manner. The algorithm has two phases: learning and testing.

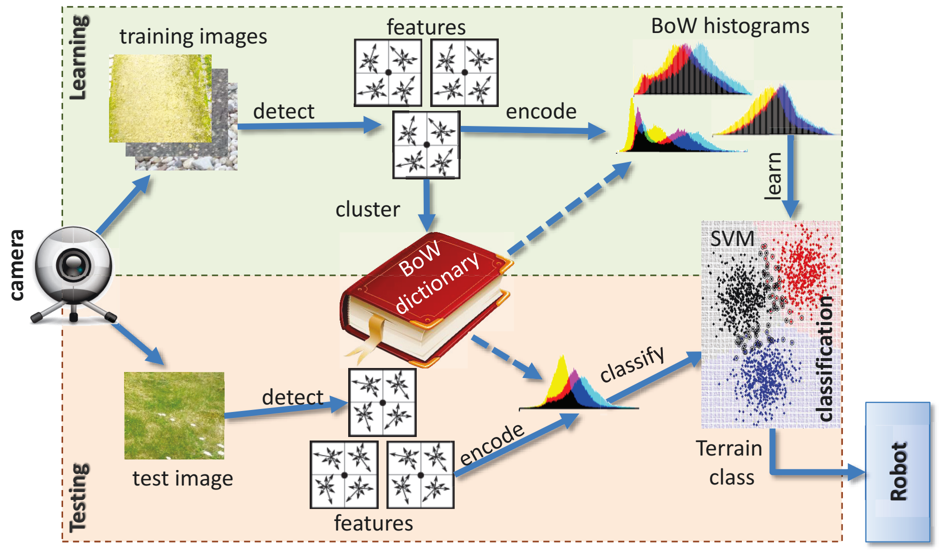


Figure 6.1: Overview of our visual terrain classification method. Upper and lower parts indicate the data flow in the training and learning phases, respectively.

The learning phase is performed off-line where all images are first stored in the memory. Data flow in this phase is indicated by arrows as shown in the upper part of Figure 6.1.

It is initiated by the extraction of local features. Here, we apply the known Speeded Up Robust Features (SURF) to generate robust local feature descriptors, details of which are provided in Section 6.2. SURF extracts various features which are to some degree invariant to scale, rotation, viewpoint, and illumination changes. However, due to differences in perspective or occlusion the number of extracted features has a large variance even for similar terrain images.

The extracted descriptors are independent and hence can be treated as a *bag of words* (BoW) in the image. This BoW nomenclature is derived from text classification algorithms in classical machine learning [He et al., 2014]. Hence, we apply the BoW method to generate vocabulary from the extracted features. In this method, all features are collected and clustered with the *k*–`means` unsupervised learning algorithm (described in Section 6.3) which yields a comprehensive BoW dictionary with a certain size.

The detected features of each image are encoded with the closest visual words in the dictionary. Next, a corresponding histogram which shows the frequency of each visual word in images is computed. The training loop is completed by SVM (its description is given in Section 5.2), essentially maximizes the geometric margin between the computed histograms.

The testing phase of terrain images includes the same steps. Given a query image again, local features are detected, encoded referring to the dictionary, and finally converted into a histogram. As the last step, the histogram is used to get a prediction from the previously computed classifier. Each descriptor is mapped to its visual word equivalent by finding the nearest cluster centroid in the dictionary.

As indicated in the lower part of Figure 6.1, this testing phase is performed when a new terrain image is acquired directly with the camera. Its output providing terrain information is further sent to higher layers of the robot control for adapting the robot behaviors to terrain.

In the next sections, we present the details of the main components of the visual classification system.

6.1 Image Descriptors

The purpose of a descriptor is to provide a unique and robust description of a feature, a descriptor can be generated based on the area surrounding an interest point.

There are many approaches for texture or pattern recognition in literature, some of which consist of visual features (Bishop,2006; Duda et al., 2001; Deselaers, 2003) and others of lidar features (Castano et al.,2001; Castano and Matthies, 2003). Some other techniques try to fuse multiple inputs to get a better performance (Fruh and Zakhor, 2003, 2002) and some use various clustering techniques (Giguere and Dudek, 2008). Pattern recognition is a very important field in computer science and finds its use in many different areas (Nabney, 2002). Following are the interest point based image descriptors that were used in this work.

6.2 Speeded Up Robust Features (SURF)

Speeded Up Robust Features (SURF) [Bay et al., 2008] are an extension of the famous Scale-Invariant Feature Transform (SIFT) [Lowe, 2004]. SURF is used to detect interest points in an image and to represent them using a 64- or 128-dimensional feature vector. These features can then be used to track the interest points across images and thus prove suitable for localization tasks. In SURF interest points are detected across the image using the determinant of the Hessian matrix. Box filters of varying sizes are applied to the image to extract the scale space. Then the local maxima are searched over space and scale to determine the interest points at the best scales. The key-point extraction capabilities of SURF, however, have been omitted. This is because the interest points detected by SURF are usually concentrated around sharp gradients, which are likely not present within homogeneous terrain patches. Instead we manually choose the interest point location and scale from which the SURF descriptor is determined. This renders our approach much faster.

The SURF descriptor is based on Haar wavelet responses and can be calculated efficiently with integral images. SIFT uses another scheme for descriptors based on the Hough transform [Oyallon and Rabin, 2013]. Common to both schemes is the need to determine the orientation. By determining a unique orientation for an interest point, it is possible to achieve rotational invariance. Before the descriptor is calculated the interest area surrounding the interest point are rotated to its direction.

For orientation assignment, SURF uses wavelet responses in horizontal and vertical direction for a neighbourhood of size $6s$. Adequate guassian weights are also applied to it. Then they are plotted in a space as given in below image. The dominant orientation is estimated by calculating the sum of all responses within a sliding orientation window of angle 60 degrees. Interesting thing is that, wavelet response can be found out using integral images very easily at any scale. For many applications, rotation invariance is not required, so no need of finding this orientation, which speeds up the process. SURF provides such a functionality called Upright-SURF or U-SURF. It improves speed and is robust for rotations upto $\pm 15°$ without performing an orientation assignment.

In our approach we divide the image in a grid and use the generated patches or subwindows to calculate the descriptors. Each image patch is then classified individually.

The SURF descriptor describes how the pixel intensities are distributed within a scale dependent neighborhood of each interest point. Haar wavelets are used to increase robustness and speed over SIFT features. First, a square window of size 20σ is constructed around the interest point, where σ is the scale of the descriptor. The descriptor window is then divided into 4×4 regular subregions. Within each subregion, Haar wavelets of size 2σ are calculated for 25 regularly distributed sample points. If x and y wavelet responses are referred by dx and dy respectively, then for the 25 sample points,

$$v = \left[\sum dx, \sum dy, \sum |dx|, \sum |dy|\right] \tag{6.1}$$

are collected. Hence, each subregion contributes four values to the descriptor vector resulting in a final vector of length 64 ($4 \times 4 \times 4$).

6.3 k–`means` clustering

Clustering is the process of partitioning a set of n observation data points $(\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n)$ where each $\mathbf{x} \in\subset R^d$ into a small number of k clusters. The goal is to assign a cluster to each data point from the given set of clusters $C = \{c_1, c_2, \cdots, c_k\}$, $(k \leq n)$ so as to minimize the within-cluster sum of squares by finding the positions $\mu_i, i = 1 \cdots k$ of the clusters that minimize the distance from the data points to the cluster. In other words, its objective is to solve:

$$\arg\min_{C} \sum_{i=1}^{k} \sum_{\mathbf{x} \in c_i} \|\mathbf{x} - \mu_i\|^2 \tag{6.2}$$

where μ_i is the mean of points in c_i. c_i is the set of points that belong to cluster i and $\|\mathbf{x} - \mu_i\|^2$ is the square of the Euclidean distance. This problem is not trivial (in fact it is NP-hard [Arthur and Vassilvitskii, 2007]), so the k–`means` algorithm only hopes to find the global minimum, possibly getting stuck in a different solution.

The k–`means` algorithm is used to solve this clustering problem and works as given in Algorithm. 1.

The algorithm eventually converges to a point, although it is not necessarily the minimum of the sum of squares [Bubeck et al., 2012]. That is because the problem is non-convex and the algorithm is just a heuristic, converging to a local minimum. The algorithm stops when the assignments do not change from one iteration to the next.

6.3.1 k–`means++`

As already described, the k–`means` method starts with a random set of k centers. In each iteration, a clustering of X is derived from the current set of centers. The centroids of these derived clusters then become the centers for the next iteration. The iteration is then repeated until a stable set of centers is obtained. The iterative portion of the above method is called *Lloyd's iteration*. This procedure will always terminate, but in practice it is common to set a maximum allowable number of iterations.

[Arthur and Vassilvitskii, 2007] modified the initialization step of k–`means` in a careful manner to provide an effective initialization procedure that decreases computation time, They obtained a randomized initialization algorithm called k–`means++` . The main idea in their algorithm is to choose the centers one by one in a controlled fashion, where the current set of chosen centers will stochastically bias the choice of the next center (given in Algorithm. 2).

Algorithm 1: k–means Clustering Algorithm

Input:
- $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$: $n \times d$ observation data points to be clustered,
- k : desired number of clusters

Output: $\{\mu_i, \cdots, \mu_k\}$: set of cluster center

```
1  Initialize the center of the clusters
2  $\{c_1, c_2, \cdots, c_k\} \leftarrow$ SelectRandomSeeds($\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}, k$)
3  for $i \leftarrow 1$ to $k$ do
4      $\mu_i \leftarrow c_i$
   /* Repeat until convergence                                          */
5  while not StoppingCriterion() do
6      for $i \leftarrow 1$ to $k$ do
7          UpdateCluster($c_i$)
       /* Attribute the closest cluster to each data point               */
8      for $j \leftarrow \arg\min_i \left\|\mu_i - x_j\right\|$ do
9          $c_i \leftarrow c_i \cup \{\mathbf{x_j}\}$                  // reassignment of vectors
       /* Set the position of each cluster to the mean of all data points
          belonging to that cluster                                        */
10     for $i \leftarrow 1$ to $k$ do
11         $\mu_i = \frac{1}{|c_i|} \sum_{x \in c_i} x$            // recomputation of centroids
```

Algorithm 2: k–means++ initialization.

```
   /* sample a point uniformly at random from X for the centroids      */
1  $c_1 \leftarrow$ SelectRandomSeeds($\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}, 1$)
2  while $|C| < k$ do
3      Sample $x \in X$ with probability $\frac{d^2(x,C)}{\phi_X(C)}$
4      $C \leftarrow C \cup \{x\}$
```

Algorithm. 2 uses a probabilistic initialization procedure. In this algorithm, the initial center $c_1 \in C$ is chosen uniformly at random from $\mathbf{x}$ and subsequent points c_i are chosen with the probability:

$$P(c_i = x \in X) = \frac{d^2(x, C)}{\phi_X(C)} \tag{6.3}$$

where $d(x, C)$ is the distance between x and the nearest center c_j that has already been chosen $(0 < j < i)$. $\phi_X(C) = \sum_{x \in X} d^2(x, C)$ is the potential function.

The advantage of this approach, besides the decrease in the overall computation time, that it tends to select centers that are further away from the ones that have already been chosen. That keeps a high variance within the centers, but maintain some probability of not always choosing the furthest point.

6.4 Feature Extraction

The main purpose of the feature extraction process is to automatically identify key points at unique locations on the image. This must be done robustly as the same feature should always be detected irregardless of viewpoint.

These *interest points* (such as areas of high contrast, T-junctions, corners, and blobs) are selected and then the neighborhood around each point is used to compute the descriptor. These descriptors should be unique and independent of the features scale or rotation. SURF relies on a fast-Hessian detector and selects a region around each key point to compute the descriptor. The descriptor is created using the Haar wavelet response in the horizontal and vertical direction.

In the standard SURF implementation of feature detection, the Hessian threshold is a constant parameter. This constant threshold value results in a widely variable number of generated key points (especially for images of varying frequency content). Figure 6.2 shows the SURF key points that are generated with the same SURF parameters and the exact threshold setting. As can obviously seen, the feature count varies too much as the algorithm generated 576 key points for the right image (grass) while it gives 2484 for the right one (gravel). Tuning the Hessian threshold parameter too high may lead to a feature description with very little data. A too low threshold will flood the classifier with redundant information and noise.

To find the threshold that produces a desired key point count, τ_d, We propose a gradient descent– like threshold adjuster. The standard gradient descent method can be viewed as a minimization of a function [Fawzi et al., 2014] as:

$$\begin{aligned} \mathbf{x_{k+1}} &= \arg\min_x f(\mathbf{x_k}) + (\mathbf{x} - \mathbf{x_k})^\intercal \nabla f(\mathbf{x_k}) + \frac{1}{2\lambda} \|\mathbf{x} - \mathbf{x_k}\|^2 \\ \text{s.t.} \quad \mathbf{x} &= \mathbf{x_k} - \lambda \nabla f(x_k), \qquad \forall k : k \geq 0. \end{aligned} \tag{6.4}$$

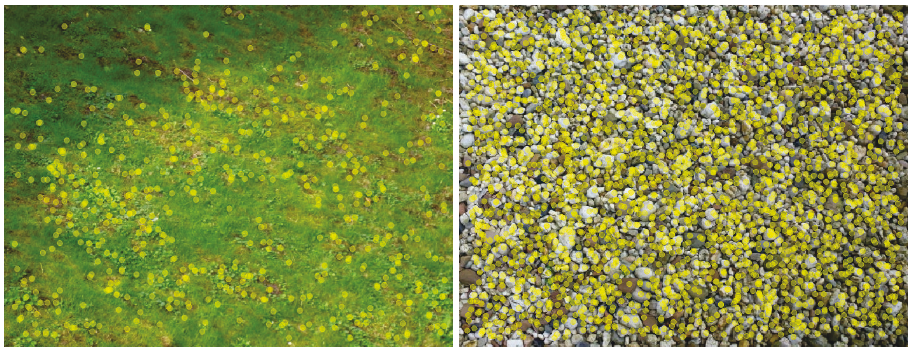


Figure 6.2: SURF key points for two images that are generated with the same SURF parameters. The yellow circles in both images are the extracted features. Key points count varies too much (left image has 576 features while the right has 2484) when using a constant Hessian threshold.

From this formulation, we implemented the following iterative convex optimization:

$$\tau_{k+1} = \tau_k + \lambda \frac{\psi(\tau_k) - N_d}{|\tau_k - \tau_{k-1}|} \tag{6.5}$$

where $\psi(\tau)$ is a continuous monotonically decreasing function that returns the number of key points for a given Hessian threshold τ and N_d is the desired number of features. k and λ are the iteration counter and step size, respectively. We use an update method similar to gradient descent, but instead of computing the function's gradient we use the error value $\psi(\tau_k) - \psi(\tau_d)$ divided by the local rate of changes $|\tau_k - \tau_{k-1}|$.

6.5 Generating a Vocabulary

To generate a vocabulary, we use the k–`means` clustering algorithm as described in Section 6.3 with the k–`means++` initialization procedure outlined in Section 6.3.1. In the context of this work, the k-means problem is to find the set of centers that best describe groupings of descriptor data.

More formally: let $k \in \mathbb{Z}$ be the desired number of clusters and let the set of n descriptors $X = \{\mathbf{x_1}, \cdots, \mathbf{x_n}\}$, $X \subset \mathbb{R}^n$. The goal is to find the k cluster centers (descriptors) in $C = \{c_1, c_2, \cdots, c_k\}$, $(k \leq n), C \subset \mathbb{Z}$ so as to minimize the potential function,

$$\phi = \sum_{\mathbf{x} \in c_i} \|\mathbf{x} - \mu_i\|^2 \tag{6.6}$$

6.6 Terrain Image Classification

Once a visual vocabulary is created the image can be described by a word frequency vector. First, a vocabulary word $\mathbf{v_i}$ is assigned to each descriptor $\mathbf{d_j}$ in the image by choosing the i that minimizes $\left\|\mathbf{v_i}-\mathbf{d_j}\right\|$. This process essentially approximates each descriptor with the vocabulary word that has the nearest Euclidean distance. Each word is then counted and the frequency of each word is stored in a vector $\mathbf{q} \in \mathbb{Z}^n$ where n is the number of visual words.

These images (each has a corresponding frequency vector) are collected in a training set and used to train the SVM classifier described in Section 5.2 using Model. 2 that employes a linear kernel as detailed in Section 5.2.1. The training set consists of k points $(\mathbf{q_i}, y_i)$ where y_i is a terrain type label assigned to determine if the word frequency vector belongs to the given class.

The training goal of the linear SVM is to classify the image based on its BoW. In Our investigation, we considered the linear kernel for the SVM due to its simplicity and computational efficiency in training and classification.

6.7 Heterogeneous Terrain Image Classification

To classify a heterogeneous terrain image (an image that contains two or more different terrain types), it is necessary to generate features with an approximately constant density across all pixels. The approach we have used for our solution is inspired by the technique presented in [Khan et al., 2012].

We start by dividing the image into cells of a given size then the cells created by this grid are used for classification. The grid size can be adjusted freely and according to the intended application. As with bigger cell size, we will have more information, it can be used for applications that require just an overview of the terrain. On the other hand, applications that require detailed terrain view can use smaller cells. This grid size parameter is a compromise between computational load and output accuracy. As smaller cells require more calculations but give a more detailed and accurate output. On the other hand, larger cells are easier to do segmentation and thus less computational load.

For classification, feature extraction is applied on each cell independently. The feature sets for each cell are then combined to form a single feature set. Afterwards, points on the image are selected on a constant grid and classification is performed in each neighboring region. This keypoint based descriptors focus on a single pixel and observe the pixels around this pixel to calculate the descriptor. At each point we iteratively resize a circle so that it encompasses the target number of features.

Once a suitable number of features is encircled, a word histogram vector is generated, and classification is performed using the linear SVM classifier trained on homogeneous terrain images. All pixels exclusively in the circle are labeled with the classification result and this step is repeated about each point. Afterwards a voting procedure is applied to each pixel by tallying the number of votes for each class. This procedure is outlined in pseudo code in Algorithm 3.

Algorithm 3: Heterogeneous Terrain Classification

Input:
- input image: $I_{m\times n}$,
- set of all terrain labels: L,
- range of desired features within radius: $\rho = \{\rho_{min}, \rho_{max}\}$.

Output: classification result matrix: $R_{m\times n}$

```
1  initialization
2    $r \leftarrow r_o$
3    $F \leftarrow$ ExtractGridFeatures($I, s_a$)
4    $C \leftarrow$ GenerateClassificationPoints($I, s_b$)
5  for $\forall c \in C$ do
6      $(F_{c,r}, r) \leftarrow$ FindRegionalFeatures($C, F, \rho, r$)
7      $l \leftarrow$ Classify($F_{c,r}$)
8      $V_l \leftarrow$ Vote($V_l, c, r$)
9  for $\forall l \in L$ do
10     $R \leftarrow$ CountVotes($V_l, R$)
```

"You have brains in your head. You have feet in your shoes. You can steer yourself in any direction you choose. You're on your own, and you know what you know. And you are the guy who'll decide where to go."

— Dr. Seuss, (American writer and cartoonist, 1904–1991)

7 Path Planning and Following

7.1 Path Planner

A path planner is used to direct the robot from its starting position through a collision-free path amidst obstacles to reach its desired destination [Burgard and Hebert, 2008].

Here, it is desired to have the path planner depends not only on the geometry of the terrain (traversability) but also on the the terrain class and characteristics predicted so far by the terrain classification modules. For this reason we selected the E* path planner [Philippsen et al., 2007].

The E* algorithm is a path planner which supports dynamic replanning and path cost interpolation, resulting in lightweight repairing of plans and smooth paths during execution. It is potential field method that interprets navigation functions as the crossing-time map of an expanding continuous surface whose propagation takes into account traversability. It supports dynamic replanning after local path cost changes. It is very similar in functionality to the Field D* but differs mostly in terms of implementations [Philippsen, 2007].

Next, the E* planer and its attributes related to this work are presented and then its use within our architecture is explained.

7.2 E* Planing Algorithm

The E* algorithm is a grid-based path planner which is based on a weighted region approach. The environment in which the robot moves is represented as a grid where the robot position and its goal position are known. A continuous navigation function is computed for each cell, stating the path's cost to reach the goal from this cell. This navigation function corresponds to a wavefront propagating from the goal toward the robot. Finally, gradient descent (Equation 6.4) over the navigation function is applied to find the path to reach the goal. Calculating the navigation function is like graph search: It starts at the goal location(s) and propagates through the grid until a path is found. By gradient descent, the robot monotonically reduces the *height* of its location on the grid just as it decreases the distance to the goal. Thus, we interpret the navigation function as a sampling of an underlying distance function.

7.2.1 E* Grid Based Environment Representation

The path-planner requires a map of the environment and the robot to be aware of its *location* with respect to the map. We assume that the robot is able to localize itself, is equipped with a map, and capable of avoiding temporary obstacles on its way. The robot's configuration space is approximated by a bounded two-dimensional grid containing obstacle information. It is made up of square cells which are either free or occupied, and the robot is considered a point on this grid. It is further assumed that the robot configuration is constrained to the center points of the cells (this is not strictly necessary).

Each E* grid cell c_i has meta information associated with it. The meta information of the cell $J(c_i)$ represents the cost[1] of traveling over this cell. It corresponds to the wavefront speed of the navigation function as:

$$J(c_i) = \begin{cases} 0 & c_i \in \text{free space} \\ \infty & c_i \in \text{obstacle} \end{cases} \tag{7.1}$$

Then for a given robot and goal positions, E* computes the navigation function for each c_i based on its $J(c_i)$. The cells corresponding to obstacles block the navigation function propagation and the cells in the neighborhood of those have their propagation cost increased. This pushes the robot away from the obstacles without blocking the navigation function propagation. To reach the goal, a trajectory is then computed using the gradient descent on the navigation function, from the robot position, to the goal position.

7.3 Terrain Dependent Navigation

In this work, the E* grid cell propagation cost is computed based on traversability, as well as on the predicted terrain type. Thus, the combined cost value can be we calculated as:

$$J(c_i) = \frac{1}{\tau \cdot \Lambda} \qquad \tau \in \{0,1\}, \Lambda \in [0,1] \tag{7.2}$$

where τ is the traversability value that depends mainly on the geometry of the terrain. Although τ usually takes a continuous, here it is considered as a Boolean value as the

[1] also called risk or difficulty

focus here is on the robot-terrain interaction so it is enough to decide if the robot to traverse the cell or not. So τ is defined as:

$$\Lambda_i = \Delta(C_i) \tag{7.3}$$

$$\tau(c_i) = \begin{cases} 1 & c_i \text{ traversable,} \\ 0 & c_i \text{ not traversable} \end{cases} \tag{7.4}$$

Λ is the *Metric of Terrain Class Navigation* (MTCN). This metric is computed based on the predicted terrain type (e.g. grass, and gravel)using the mapping function $\Delta(C_i)$ that converts the terrain class C_i of the cell c_i to its metric value. According to this metric definition, it takes a high value for a terrain having a *good* robot interaction. Hence, The robot has a *perfect* behavior on a terrain with $\Lambda = 1$. On the other hand, a terrain with $\Lambda = 0$ corresponds to a terrain to be avoided.

In a typical application, the robot and goal positions are given to E* on the environment map. For example in the configuration the depicted in Figure 7.1.

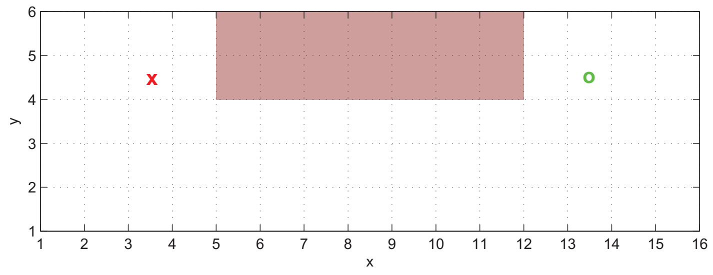


Figure 7.1: Simple E* path planner example. The navigation map divided into grid cells with two terrains: pavement (white) and gravel (brown) depicted on the environment map with $\Lambda_p > \Lambda_g$. The robot and goal positions are marked with the (x) and (o), respectively.

In this configuration, the robot, in its current position which is marked by the cross, has to reach the goal on the right hand side (marked by circle). The robot is assumed to correctly classified the two terrains around it as pavement (the white area) and gravel (the gray area). If the robot is able to identify the MTCN of the two terrains according to:

$$\Lambda_p > \Lambda_g \tag{7.5}$$

where Λ_p and Λ_g are the MTCN for pavement and gravel, respectively. Assuming that both terrains are traversable (i.e. $\tau = 1$), we substitute into Equation 7.2 to get the combined cost values. That leads to:

$$J(c_p) < J(c_g) \tag{7.6}$$

The navigation function is then computed for each cell c_i based on its $J(c_i)$. The cells corresponding to obstacles block the navigation function propagation and the cells in the neighborhood of those have their propagation cost increased (see Figure 7.2(top)).

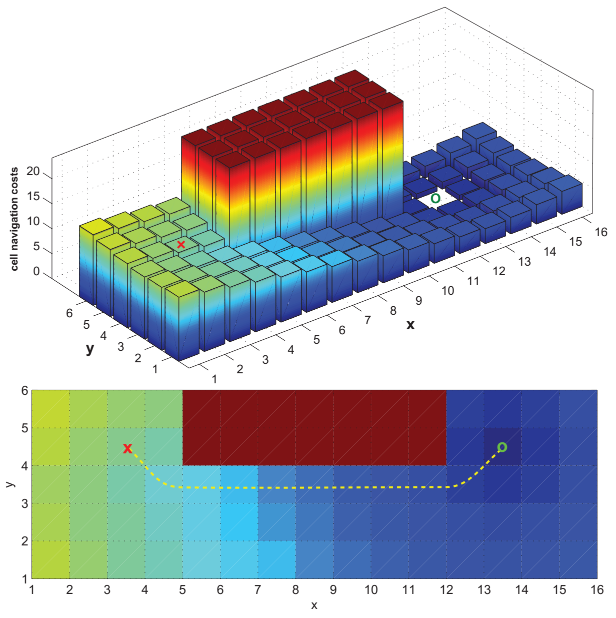


Figure 7.2: Simple E* path planner example calculations. Wavefront propagation from the goal and and calculated navigation function costs for each cell (top) and The resulting generated path (yellow dashed) provided by E* from the robot position to the goal (bottom).

This pushes the robot away from the obstacles without blocking the navigation function propagation. The trajectory reaching the goal can be then computed using the gradient descent on the navigation function, from the robot position, to the goal position. The resulting generated path (Figure 7.2(bottom)) provided by E* naturally avoids the gravel terrain and is a result of a gradient descent performed on the navigation function.

7.4 E* Path Calculations

Our combined navigation path planning problem can be stated as:

> Given a waypoint at distant (l) from the current robot position, If the direct trajectory to the goal is over a terrain with a specific MTCN (Λ_1) while there is a neighboring terrain with a better MTCN (Λ_2). Find the maximum deviation (d), the rover should be allowed to take traversing the better terrain to optimize the navigation costs to reach the goal.

This problem is depicted in the schematic in Figure 7.3. In this setup, the two terrains represent two different classes (namely C_1 and C_2 with their associated navigation costs. Terrain of C_2 is assumed to have better MTCN. Therefore, the relative propagation costs can be expressed as:

$$J_2 = \frac{1}{\alpha} J_1, \quad \alpha > 0. \tag{7.7}$$

where α is the relative value of propagation cost. The deviation distance (d) from the direct path is expressed relative to the path length l as:

$$d = \frac{1}{\beta} l, \quad \beta > 0. \tag{7.8}$$

The two ratios, α and β are dependent on the direct path distance l and the deviation distance (d). This dependency is further investigated next.

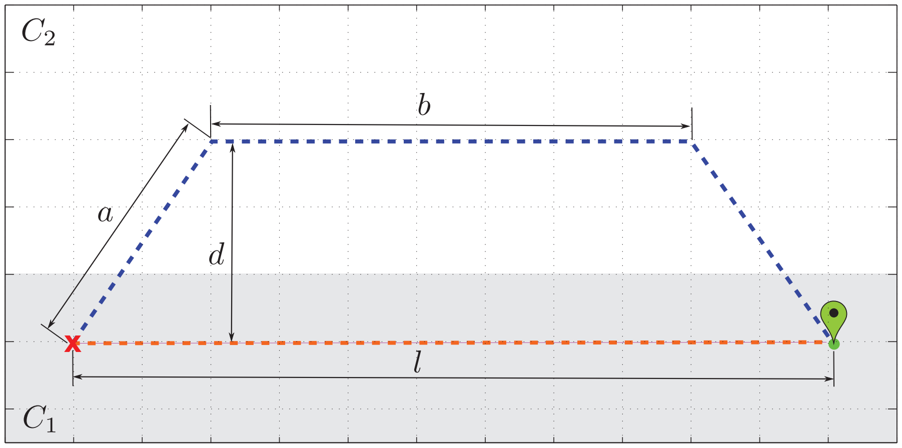


Figure 7.3: Schematic for E* path planner calculations. The two terrains pavement (C_2, white) and gravel (C_1, gray) with $\Lambda_p > \Lambda_g$ are depicted on the environment map. The robot and goal positions are marked with the (x) and (o), respectively.

As shown in Figure 7.3, the cost of traveling along the direct path can be expressed as:

$$J_d = l\, J_1 \tag{7.9}$$

While the cost of following the alternative path is:

$$J_a = 2a\, J_1 + b\, J_2 \tag{7.10}$$

The segment a can be expressed as:

$$a = \sqrt{(l-b)^2/4 + d^2} \tag{7.11}$$

The segment b can be expressed relative to the direct path length as:

$$b = \gamma\, l \tag{7.12}$$

Substituting with Equation 7.7 through Equation 7.12 into Equation 7.10, we finally get the expression of the alternative path cost as:

$$J_a = 2\,J_1\sqrt{(l-b)^2/4+d^2}+\gamma\,l\frac{J_1}{\alpha} \tag{7.13}$$

To get the optimal length of the segment b, we find the maximum of J_a with respect to γ as:

$$\frac{\partial J_a}{\partial \gamma} = \frac{l^2(\gamma-1)}{2\sqrt{l^2(1-\gamma)^2/4+d^2}}\,J_1+\frac{l}{\alpha}\,J_1 = 0 \tag{7.14}$$

After mathematical manipulation with Equation 7.14 and substituting with β as defined in Equation 7.8, we reach the limiting value of γ as:

$$\gamma = 1-\frac{2}{\beta\sqrt{\alpha^2-1}} \tag{7.15}$$

At the limiting value of γ given by Equation 7.15, the costs of the two paths are equal. Hence:

$$J_d = J_a \tag{7.16}$$
$$l\,J_1 = 2a\,J_1+b\,J_2 \tag{7.17}$$

substituting for J_2 as given in Equation 7.7 and using the previous relations, we finally have:

$$\frac{\gamma}{\alpha}+2\sqrt{(1-\gamma)^2/4+d^2/l^2} = 1 \tag{7.18}$$

Substituting for γ as given in Equation 7.15, we have:

$$\frac{1}{\alpha}\left(1-\frac{2d}{l\sqrt{\alpha^2-1}}\right)+2\sqrt{\frac{d^2}{l^2(\alpha^2-1)}+\frac{d^2}{l^2}} = 1 \tag{7.19}$$

Thereafter, substituting with β as defined in Equation 7.8 and simplifying the result we have:

$$\alpha = \frac{\beta^2+4}{\beta^2-4} \tag{7.20}$$

This relation sets the limit to decide selecting the alternative path over the direct path based on their relative MTCNs. As plotted in Figure 7.4, the asymptote at propagation cost ratio ($\alpha = 1$) corresponds to both terrains having the same MTCN. Hence, no advantage will be gained from traveling the alternative path. hence, the robot is to take the direct one. On the other hand, the one at ($\beta = 2$) is for the deviation distance to direct path ratio which is for the robot to just take the alternative path over the better terrain then immediately back to the reach the goal.

From these remarks, the limit between terrain class prediction actively influencing the robot trajectory or not are given as:

$$\alpha > 1, \quad \beta > 2. \tag{7.21}$$

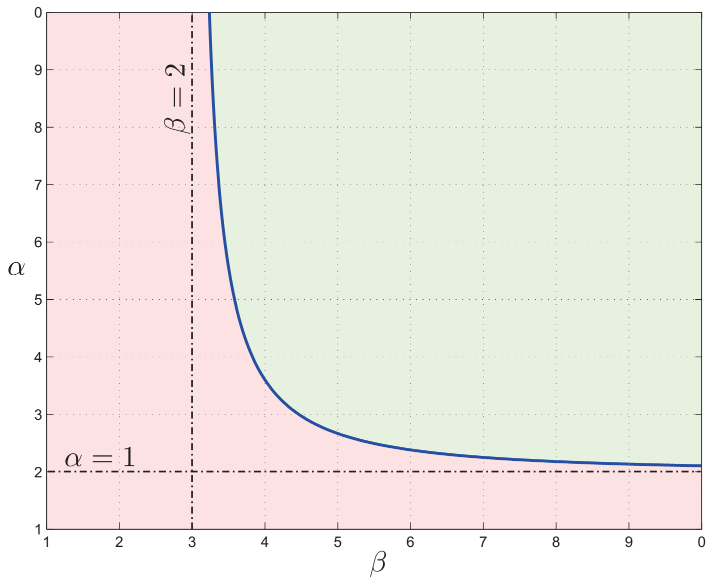


Figure 7.4: Decision ranges for alternative path selection. The two parameters decide selecting an alternative path to target (green) instead of direct one (red) influenced by terrain types. The asymptote at propagation cost ratio ($\alpha = 1$) corresponds to both terrains having the same MTCN while the one at the deviation distance to direct path ratio ($\beta = 2$) is just for going to the alternative terrain then immediately back.

7.5 Path Cost Calculations

The navigation cost is a value defined between 0 and 1 (as defined in Equation 7.2). Using E* approach, the environment grid is initialized with 1, corresponding to the fastest wave propagation. Each time a terrain with a new class is identified, its corresponding environment cells are updated with its MTCN and the desired navigation path is adapted accordingly.

For unknown terrain types (i.e., the ones classified as unknown), The MTCN of these terrains cannot be computed. Thus several strategies could be used:

Explore: the unknown terrain is considered as better, or as good as the best MTCN ($\Delta_i = 1$). The result of such a strategy will set the robot to favor unknown areas and hence go looking for them. This strategy can be risky in some situations.

Avoid: here, this terrain MTCN is set to the worst ($\Delta_i = 0$). This results in the robot to systematically avoid unknown terrains. This is a very conservative strategy that is useful if safety was of prime importance.

In our work, we adopted an intermediate strategy to balance the two opposite Explore and Avoid strategies. In our strategy, if the area of the unknown terrain is less than half of the robot size, the robot operate in the Explore mode. In this mode the MTCN of the unknown class is set to the mean of the neighboring cells. On the other hand, if the unknown terrain class area is bigger than half of the robot size, the robot is set to avoid it by setting its MTCN to 0.

Our strategy will maintain the robot safety guaranteed due to the small size of the unknown area. As chance is very high that the robot is able to overcome that terrain in case it is unsuitable for its navigation. At the same time, the robot is allowed to gather some valuable sensor information during its interaction with that terrain.

7.5.1 Assumptions

The most important assumptions driving the implementation of the described path planner approach on the robot are summarized as:

- While numerous works are identifying the traversability (τ) of the robot surrounding, the focus here is on robot–terrain interaction. The consequence is that the traversability is not taken into account, or rather, the terrain surrounding the robot is assumed traversable (i.e., $\tau = 1$ as in Equation 7.4).
- The terrain surrounding the rover is assumed to be globally flat. As we are interested in showing the adaptive behavior of the robot, the experiments conducted uses flat grounds with varied MTCN characteristics.
- Knowledge acquired during past tests can be reused, but this is not necessarily. It is assumed that the approach is not dependent on any prior knowledge regarding the remote terrain types. Nevertheless, a predefined set of terrain classes, such as grass, gravel and so on, is required.
- The adaptive behavior is driven by MTCN (Λ) whose characteristics are assumed to be known and provided by the user. In other words, the meaning of what a *good* and *bad* terrain must be predefined and provided for each terrain class (i.e., defining the mapping function Δ in Equation 7.3).

7.6 Path Following

To enable the robot to follow a desired path (usually provided by its planer or operator), a path following controller is developed. With this controller module, we implement how the robot tracks a reference path via the control of its actuators. The implemented controller module is based on simple and effective control strategies. As applying open–loop steering control only is not robust to modeling errors (originating from numerous sources) because it cannot guarantee that the robot will move along the desired trajectory as planned. This is why our implementation is based on feedback control.

The implementation supposes that measurements of the variables involved in the control loop (typically the position and orientation of the robot with respect to either a fixed frame or a path that the robot should follow) are available. A large part of the presented approach is based on linear control theory. The feedback control laws then inherit the strong robustness properties associated with stable linear systems.

7.6.1 Robot Model

The robot kinematic model approximates its mobility. The configuration of a nonholonomic robot[2] is represented by the position and orientation of its main body in the plane, and by the steering angle [Oriolo et al., 2002]. As depicted in Figure 7.5, the two velocity inputs, forward velocity (u_1) and steering velocity (u_2), are available for motion control. The applied kinematic model for the robot approximating its mobility [Chung et al., 2008] can be expressed as:

$$\begin{aligned}\dot{x} &= u_1 \cos\theta \\ \dot{y} &= u_1 \sin\theta \\ \dot{\theta} &= u_2 \end{aligned} \tag{7.22}$$

where (x, y) represents the coordinates of the point P located at the origin of the robot frame, the angle θ characterizes the robot chassis orientation, ϕ represents the robot steering angle, and L is the distance between the point P and the robot front. In this equation, u_1 and u_2 represent the driving and the steering velocity input, respectively, as represented by the block diagram in Figure 7.6.

[2]The nonholonomic nature of the robot is related to the assumption that the robot moves without slipping

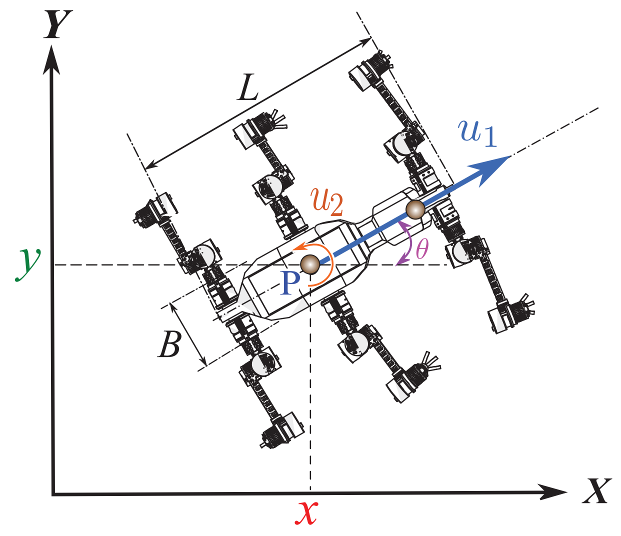


Figure 7.5: Configuration variables of the robot kinematic model

In this model, the configuration of the robot is represented by the position and orientation of its main body in the plane and the two velocity inputs are available for motion control. This model covers in a realistic way our robot.

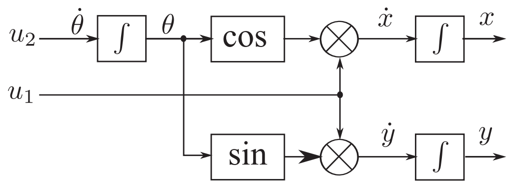


Figure 7.6: Robot kinematic model block diagram

7.7 Path Reference Frames

Here, it is implicitly considered that the controller is to be embedded in a hierarchical architecture in which a higher–level planner solves the obstacle avoidance problem and provides a series of motion goals to the lower control layer. In this perspective, the controller deals with the basic issue of converting ideal plans into actual motion execution. Wherever appropriate, the interactions between feedback control and motion planning primitives shall be highlighted, such as the generation of open-loop commands and the availability of a feasible smooth path joining the current robot position to the destination.

The analysis is limited to the case of a robot in a workspace free of obstacles. Let us consider a curve $\mathscr{C}$ in the plane of motion, as illustrated in Figure 7.7, and let us define three frames π_w, π_r, and π_s, as follows:

- $\pi_w = \{0, i, j\}$ is the world fixed frame,
- $\pi_r = \{P, i_r, j_r\}$ is a frame attached to the robot with its origin, P, located on the mid-distance of the middle axis of the robot, and
- $\pi_s = \{P_s, i_s, j_s\}$, which is indexed by the curve curvilinear abscissa s, is such that its unit vector i_s tangents $\mathscr{C}$.

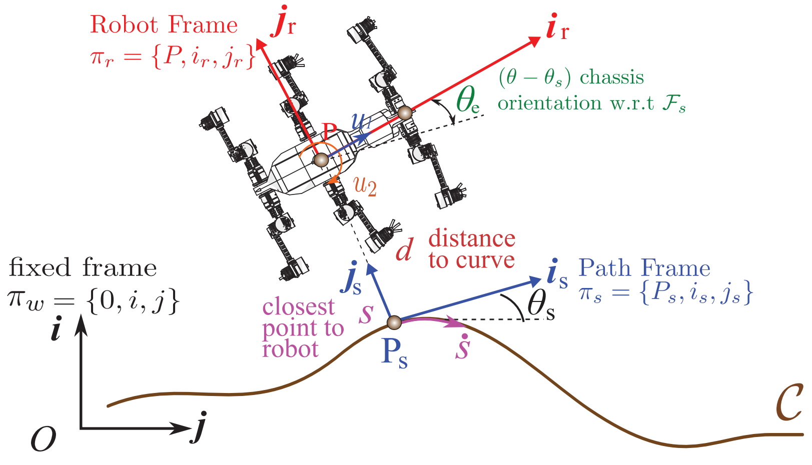


Figure 7.7: Path-following geometric reference frames

The equations of motion of P with respect to the curve $\mathscr{C}$ can be expressed based on the three variables s, d, and θ_e, defined as:

- s is the curvilinear abscissa at the point P_s obtained by projecting P orthogonally on $\mathscr{C}$. This point exists and is unique if the point P is close enough to the curve (the distance between P and the curve must be smaller than the lower bound of the curve radii as given by the initial conditions in Equation 7.28 and Equation 7.31)
- d is the ordinate of P in the frame π_s; its absolute value is also the distance between P and the curve.
- $\theta_e = \theta - \theta_s$ is the angle characterizing the orientation of the robot chassis with respect to the frame π_s.

By definition of the curvature $c(s) = \frac{\partial c(s)}{\partial s}$ of $\mathscr{C}$ at P_s, the derivative of these variables can be derived as:

$$\begin{aligned}
\dot{s} &= \frac{u_1}{1 - dc(s)} \cos\theta_e \\
\dot{d} &= u_1 \sin\theta_e \\
\dot{\theta}_e &= u_2 - \dot{s}c(s)
\end{aligned} \tag{7.23}$$

7.7.1 Kinematic Model into Chained Form

The *chained form* is one of the canonical forms for kinematic models of nonholonomic robots utilized for the systematic development of both open-loop and closed-loop control strategies [Murray and Sastry, 1991]. A block diagram representation of a general chained system is shown in Figure 7.8

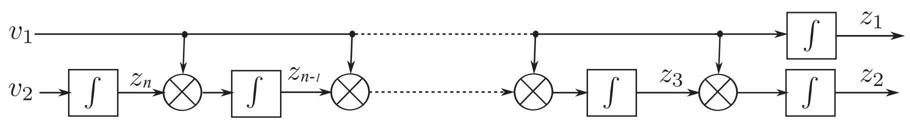


Figure 7.8: General chained form block diagram

The model given in Equation 7.23 is transformed to the chained form through the change of coordinates and control variables $(s, d, \theta_e, u_1, u_2) \mapsto (z_1, z_2, z_3, v_1, v_2)$ defined by:

$$\begin{aligned} z_1 &= s, \\ z_2 &= d, \\ z_3 &= [1 - dc(s)] \tan\theta_e, \\ v_1 &= \dot{z_1}, \\ v_2 &= \dot{z_3} \end{aligned} \tag{7.24}$$

The single-chain form represented by Equation 7.24, although nonlinear, has a strong underlying linear structure. This clearly appears when u_1 is assigned as a function of time, and is no longer regarded as a control variable. In this case, Equation 7.24 becomes a single-input, time-varying linear system as illustrated with the block diagram in Figure 7.9.

To derive the two control inputs (v_1 and v_2), Equation 7.23 is used to get:

$$v_1 = \dot{z_1} = \dot{s} = \frac{u_1}{1 - dc(s)} \cos\theta_e \tag{7.25}$$

From (Equation 7.24), the derivative of z_3 is needed for v_2, so first, we break out z_3 as:

$$\begin{aligned} v_2 = \dot{z_3} = &-[\dot{d}c(s) + d\frac{\partial c}{\partial s}\dot{s}] \tan\theta_e \\ &+ [1 - dc(s)](1 + \tan^2\theta_e)\dot{\theta}_e \end{aligned} \tag{7.26}$$

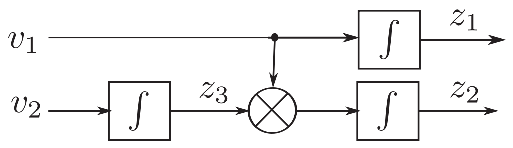


Figure 7.9: Robot chained form block diagram

7.8 Path Following with Orientation Control

We reconsider the path-following problem with the objective to synthesize a control law which allows the robot to follow the path in a stable manner. The control law should asymptotically stabilize ($d = 0$, $\theta_e = 0$) and also ensures that the constraint on the distance d to the path (i.e., $|dc(s)| < 1$) is satisfied along the trajectories of the controlled system.

7.8.1 Proportional Input-Scaling Controller

We use a standard linear control design that provides convergence provided that the robot starts sufficiently close to the desired trajectory. Let k_2, k_3 denote parameters such that the characteristic polynomial of the closed loop system is Hurwitz stable (i.e., when $k_3, k_2 > 0$). With these parameters we can construct the proportional-feedback-like control law:

$$\begin{aligned} v_2 &= -v_1 \sum_{i=2}^{3} sgn(v_1) k_i z_i \\ &= -v_1 \left[sgn(v_1)^3 k_2 z_2 + sgn(v_1)^2 k_3 z_3 \right] \\ &= -v_1 k_2 z_2 - |v_1| k_3 z_3 \end{aligned} \tag{7.27}$$

The constraint $|dc(s)| < 1$ is satisfied with the initial stability criteria:

$$z_2^2(0) + \frac{1}{k_2} z_3^2(0) < \frac{1}{c_{max}^2} \tag{7.28}$$

where $c_{max} = \max_s |c(s)|$

From a practical point of view, it is useful to complement the control action with an integral term. Let us define a new state variable z_0 by:

$$\dot{z}_0 = v_1 z_2, \quad z_0(0) = 0 \tag{7.29}$$

Using this variable, we can modify the control action given in Equation 7.27 to be:

$$v_2 = -|v_1| k_0 \int_0^t v_1 z_2 - v_1 k_2 z_2 - |v_1| k_3 z_3 \tag{7.30}$$

It can be seen that the system is stable when $k_3, k_2, k_0 > 0$ and the following initial condition is satisfied:

$$z_2^2(0) + \frac{1}{k_2 - \frac{k_0}{k_3}} z_3^2(0) < \frac{1}{c_{max}^2} \tag{7.31}$$

The implemented input-scaling controller types are represented in block diagram in Figure 7.10 for the proportional and proportional plus integral variations.

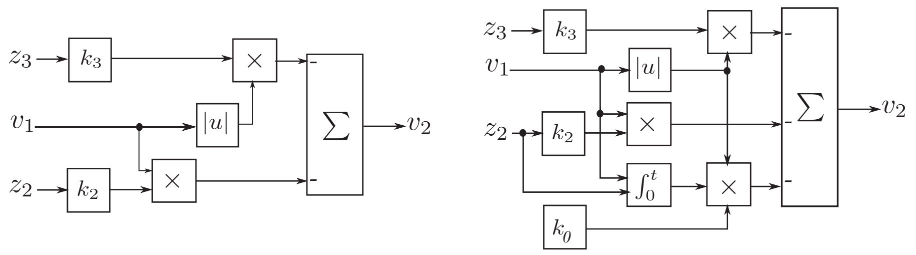


Figure 7.10: Path following controller block diagram: proportional (left) and proportional plus integral (right)

7.9 Path Following Controller Implementation

The outlined control actions are implemented and tested as a stand-alone c++ library. Then, the control library is encapsulated into a ROCK component and integrated into the robot as depicted in Figure 7.11.

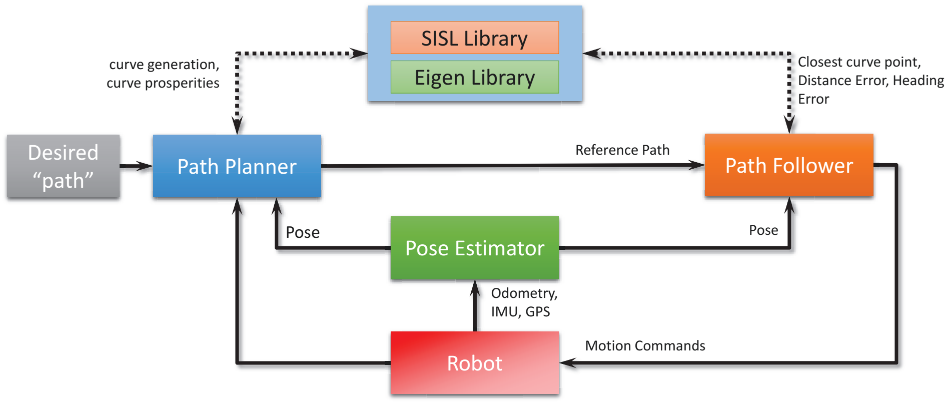


Figure 7.11: Path-following controller implementation and its connections with other components (as implemented within the [ROCK, 2015] framework).

7.10 Camera Model

Here we discuss the model of the camera used to compute the terrain patch locations from the camera image. The camera considered here is a perspective camera as it is a pinhole camera which performs perspective projection. Figure 7.12 depicts the geometry of the model with the relative position of several reference frames.

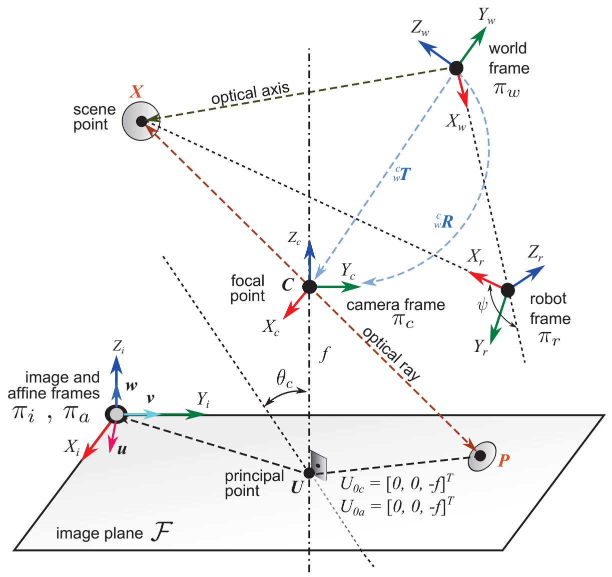


Figure 7.12: Perceptive camera model placed on a robot with related reference frames attached. The focal length f (also known as the principal axis distance) is a parameter of the lens.

The plane on the bottom $\mathscr{F}$ is an image plane to which the real world is projected, and the vertical center line is the optical axis. The lens is positioned perpendicularly to the optical axis at the focal point **C** (also called the optical center). The focal length f (sometimes called the principal axis distance) is a parameter of the lens.

The projection is performed by an optical ray reflected from a scene point **X**. The optical ray passes through the optical center **C** and hits the image plane at the point **U**. In a first, we define the following reference frames:

- π_w: world fixed reference frame. It is the fundamental reference frame whose X_w and Y_w axes are within the horizontal plan. Points **X**, **U** are expressed in this coordinate system.
- π_c: camera reference frame attached to the camera, and its X_c and Z_c axes are tilted according to the camera pitch angle. It has the focal point **C** as its origin. π_c axis Z_c is aligned with the optical axis and points away from the image plane.
- π_i: the image reference frame that has axes aligned with π_c, with its axes X_i and Y_i lying in the image plane and Z_i parallel to the optical axis.

π_a: image affine reference frame that has coordinate axes u, v, w, and origin coincident with the origin of π_i.

π_r: robot reference frame that is attached to the robot with its X_r axis pointing in the forward direction and Z_r pointing up relative to the robot body.

The reason for introducing the camera reference frame π_a is the fact that, in general, pixels need not be perpendicular and axes can be scaled differently. For that reason, π_a axes w, v are aligned with the axes Z_i, X_i, but the axis u may have a different orientation to the axis Y_i.

The camera performs a linear transformation from the 3D projective space $\mathbb{R}^3$ to the 2D projective space $\mathbb{R}^2$. A scene point **X** is expressed in π_w as a 3×1 vector. To express the same point (i.e. $\mathbf{X_c}$) in π_i, we have to rotate it as specified by the matrix **R** and translate it by subtracting vector **T**, formally:

$$\mathbf{X_c} = [x_c, y_c, z_c] = \mathbf{R}(\mathbf{X}_w - \mathbf{T}) \tag{7.32}$$

The point $\mathbf{X_c}$ is projected to the image plane $\mathscr{F}$ as point $\mathbf{U_c}$. The coordinates of $\mathbf{U_c}$ can be derived from the similar triangles illustrated in Figure 7.12 as:

$$\mathbf{U_c} = \left[\frac{-f\,x_c}{z_c}, \frac{-f\,y_c}{z_c}, -f\right]^{\mathsf{T}} \tag{7.33}$$

It remains to derive where the projected point $\mathbf{U_c}$ is positioned in π_a, i.e. to determine the coordinates which the real camera actually delivers. We use π_a, with its origin at the top left corner of the image. Here, π_a accounts for the shear and rescaling (often called the *aspect ratio*) of π_i.

The principal point[3] $\mathbf{U_0}$ is expressed in π_a as $U_{0a} = [u_0, v_0, 0]^{\mathsf{T}}$. The projected point can be represented in the 2D image plane $\mathscr{F}$ in homogeneous coordinates as $\mathbf{P} = [U, V, W]^{\mathsf{T}}$, and its 2D Euclidean counterpart is $u = [u, v]^{\mathsf{T}} = [U/W, V/W]^{\mathsf{T}}$. These homogeneous coordinates allow us to express the affine transformation as a multiplication by a single 3×3 matrix where unknowns a, b, c describe the shear together with scaling along coordinate axes, and u_0 and v_0 give the affine coordinates of the principal point in the image as:

$$\mathbf{P} = \begin{bmatrix} U \\ V \\ W \end{bmatrix} = \begin{bmatrix} a & b & -u_0 \\ 0 & c & -v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{-f\,x_c}{z_c} \\ \frac{-f\,y_c}{z_c} \\ 1 \end{bmatrix} \tag{7.34}$$

The above upper triangular matrix is called the *calibration matrix* of the camera; and the notation **K** will be used for it.

[3]sometimes called the center of the image in camera calibration procedures. It is the intersection of the optical axis with the image plane $\mathscr{F}$.

We aim to collect all constants in the camera calibration matrix **K**. Since homogeneous coordinates are used, we left multiply Equation 7.34 by z_c to get:

$$z_c\mathbf{P} = \begin{bmatrix} -af & -bf & -u_0 \\ 0 & -cf & -v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} \tag{7.35}$$

$$= \mathbf{K}\,\mathbf{R}(\mathbf{X}_w - \mathbf{T}) \tag{7.36}$$

The coefficients of the upper triangular camera calibration matrix **K** are called *intrinsic parameters* of the camera, and describe the specific camera independent of its position or orientation in space. For a camera with fixed optics, these parameters are identical for all the images taken with the camera. On the other hand, cameras with zooming and focusing capabilities, the focal length can obviously change, but also the principal point can vary.

The camera *extrinsic parameters* depend on the orientation of π_c with respect to π_w (see Figure 7.12). This relation is given in Equation 7.36 by matrices **R** and **T**. When they are known together with the intrinsic parameters[4], a metric measurement can be directly performed from the images.

If π_i is selected to have the same axes as π_c, but oriented according to π_a. The complete transformation can be expressed as:

$$\mathbf{X_i} = {}^{i}_{c}\mathbf{R} \times \left({}^{c}_{r}\mathbf{R} \times \left({}^{r}_{w}\mathbf{R} \times \left(\mathbf{X_w} - {}^{r}_{w}\mathbf{T}\right) - {}^{c}_{r}\mathbf{T}\right)\right) \tag{7.37}$$

The notation adopted in [Craig, 2004] is used here. In which ${}^{b}_{a}\mathbf{R}$ denotes the rotation matrix from π_a to π_b and ${}^{b}_{a}\mathbf{T}$ is the translation vector from π_a to π_b within π_a.

The intrinsic parameters of the camera calibration matrix **K** estimation based on the correspondence between the known position of points in π_w and their corresponding pixel values in π_i. The details of these calibrations as well as the camera parameters of the configurations used during the tests are given in Section 8.6.

7.10.1 Transforming from Image to World Coordinates

Conversion from image coordinates to world coordinates is a fundamental step for projection of the classified image patches to the world map used in the path planning step (described in Section 7.3).

To perform this transformation, first we have to know the specific calibration information of the used camera, which are essential for these conversions. These parameters include geometry (e.g. image width and height), intrinsic, and extrinsic parameters (see

[4]For most cameras the pixels are almost perfectly rectangular and thus b is very close to zero. Furthermore, the principal point is often close to the center of the image. These assumptions can often be used to get a suitable initialization for a more complex iterative estimation procedure to calibrate the camera.

Section 8.6 for a description of the experimental process to obtain them). Then, these parameters are substituted into Equation 7.36 which is rewritten here as:

$$z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \kappa_u & \kappa_s & -u_0 \\ 0 & \kappa_v & -v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} \tag{7.38}$$

where $\kappa_u = -af$, $\kappa_s = -bf$, and $\kappa_v = -cf$.

Equation 7.38 can be split into two separate equations for u and v as:

$$u = \alpha_u \frac{x_c}{z_c} + \alpha_s \frac{y_c}{z_c} - u_0 \tag{7.39}$$

$$v = \alpha_v \frac{y_c}{z_c} - v_0 \tag{7.40}$$

It can be also written for the camera frame as:

$$x_c = \frac{1}{\alpha_u}(u + u_0) z_c \tag{7.41}$$

$$y_c = \frac{1}{\alpha_v}(v + v_0) z_c \tag{7.42}$$

As we only have one view, we actually can not get the reverse of these equations unless we also know its z-axis. That is only possible if additional information is added. This information can be obtained using a complementary sensor, such as a range finder, or a correlation between two images (using a stereo camera). Here, we simply use the flat ground assumption to compute the distance z_i. In the implementation of this part, we used the available tools in [ROCK, 2015] to handle the required coordinate transformation operations.

7.11 Summary

In this chapter, the description of the robot path planning and following control and the influence of the terrain type on them are presented.

At the high level, E* algorithm is employed for path planning. E* offers a trade off between the movement cost and the path length and it provides a path to be followed to reach the goal. That is a result of applying gradient descent performed on the navigation function. This path naturally avoids obstacles and favors terrains with lower MTCN.

At the low level, a proportional input-scaling controller is designed and implemented to autonomously steer the robot to follow a desired path in a stable manner.

"It doesn't matter how beautiful your theory is, it doesn't matter how smart you are. If it doesn't agree with experiment, it's wrong."

— Richard P. Feynman, (American theoretical physicist, 1918–1988)

8 Experiments and Results

For legged robots, evaluation of the overall mobility performance on different soil types is a requirement to guarantee a good walking behavior. Therefore, it is important to develop models and simulations of the overall leg-soil interaction and terramechanical behavior, making use of semi-empirical and physical models.

To help build these interaction models and for the verification and validation of them, intensive robot testing in a lab environment is necessary and should occur hand in hand with their development.

8.1 Terramechanics Real and Simulation Testbed

To prepare for extensive experiments to study the interaction between walking robot leg and the terrain, a new concept is developed. In this concept a robotic arm (from Schunk as seen in Figure 8.2) is to be used to move an attached robot leg according to the exact motion profile of the whole real robot. The whole Schunk arm can be viewed in this concept as the whole robot starting from the shoulder joint and up. The leg movement trajectories are generated using the robot motion planner through the robot arm control library (described in Section Section 8.1.3).

This section describes the design and development of a new testbed for legged robot systems. This testbed consists of a soil testbed, a 7DoF robotic arm, and software architecture to perform both real and simulated experiments with different robot legs and other test objects.

The testbed is used to study and analyze the test objects interaction with the soil and their terramechanical properties to help build models that are used in the design of robots and in the overall simulations of these systems. The developed robot leg terramechanics testbed consists of three main components, namely, the real testbed, control software, and simulator. Figure 8.1 depicts an overview of the whole system and a detailed description of these components is given in the following subsections.

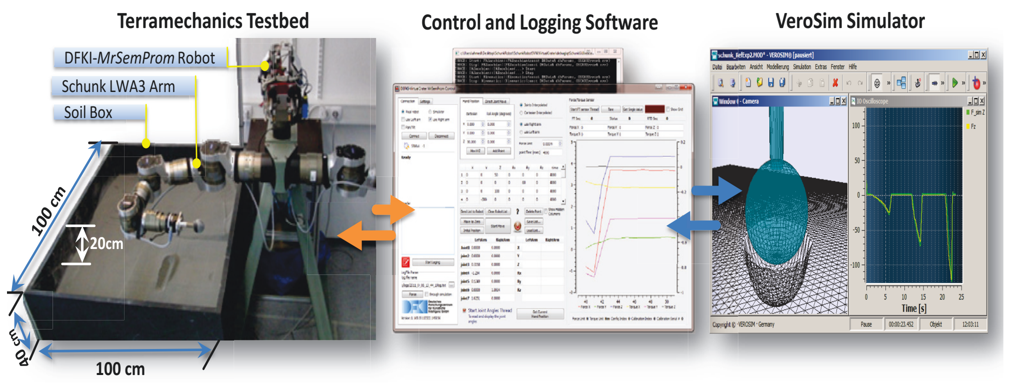


Figure 8.1: Robot foot terramechanics testbed architecture overview with its main components, namely, the real testbed, control software, and simulator.

8.1.1 Terramechanics Soil Box

The developed soil testbed setup, depicted in Fig. Figure 8.1 consists of a wooden soil box of 100 cm width, 100 cm length and 40 cm height, filled up to 20 cm with soil. This soil box is large enough to avoid side wall and bottom effects in the system level test runs for robot legs within the dimension of the actual robot leg prototypes. It is also possible to divide the soil box into two parts to cope with two different types of soft soils.

8.1.2 Schunk Arm

To have a realistic walking trajectory as that in the walking real robot, an approach in which the test leg is attached to a seven degrees of freedom (7DoF) robotic arm of our DFKI–MrSemProm robot is implemented (shown in Figure 8.1).

MrSemProm [de Gea Fernández et al., 2009] dual arms are constructed with two Schunk LWA3 manipulators. Each arm has a combination of high power density servo-electric swivel (brushless servomotors with harmonic drive power trains in addition to integrated motor–controller unit) modules (namely PRL modules) and light interconnection materials (aluminium structure for weight optimization). The PRL modules have standard connecting elements in a 7-axis structure (2x PRL 120, 2x PRL 100, 2x PRL 80, 2x PRL 60) that provides the 7DoF capabilities.

To precisely measure the forces and torques generated by the contact between the robot leg and the soil, an ATI Net F/T force/torque sensor [ATI URL, 2014] is used. This sensor ensures precise measuring of forces and torques in all three spatial directions (six components; three for force and three for torque), high degree of resolution of measured values and fast data transfer for virtually real–time force monitoring. The force/torque sensor is located between the robot leg and MrSemProm arm end–effector (Figure 8.4).

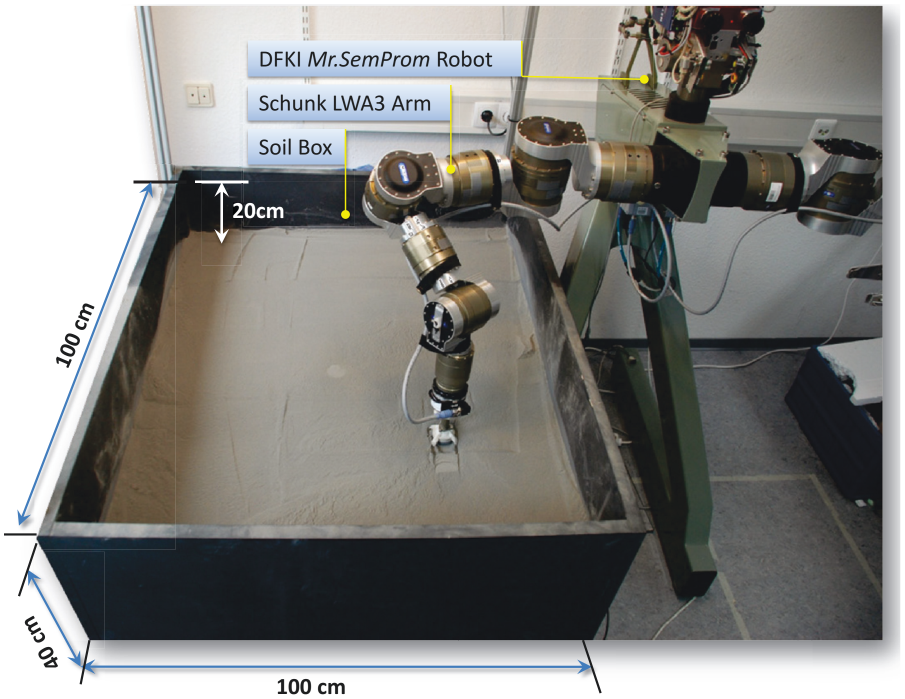


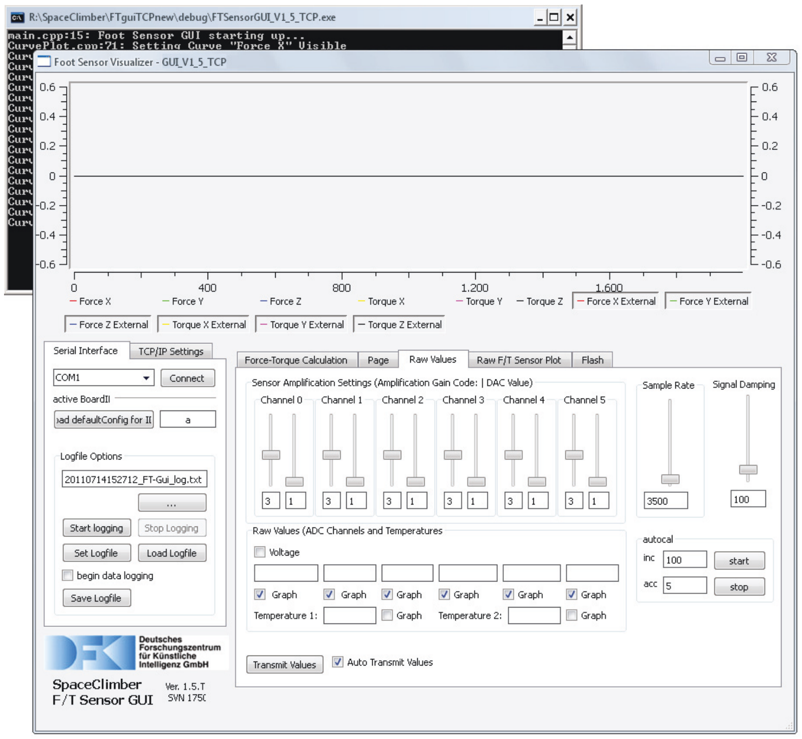
Figure 8.2: Terramechanics testbed with a test robot foot attached to the end effector 7DoF Schunk arm of of DFKI–MrSemProm robot.

A dedicated software library was developed to handle setting up, synchronizing, and logging the sensor data. the GUI of this Library is shown in Figure 8.3.

Additionally, to facilitate connecting different foot modules to the robot arm, different mechanical fixture adapters are fabricated and attached to the very end of the end effector (details of one of these adapters is shown in Figure 8.4)

8.1.3 Control Software

The control software is developed to control DFKI–MrSemProm arm for a desired trajectory of the attached test–leg. This software is split into a library utility (in a DLL/Lib component) that uses multithreading and functions to send control commands to the robot joints and a user interface GUI that interacts with the high level behaviours and commands to control the robot, to access the force/torque sensor, and logging of the experiment data. The front end of this program is shown in Figure 8.1-(middle). The main tasks of these software components and the interaction among them and the robot are shown in Fig Figure 8.5.



```
main.cpp:15: Foot Sensor GUI starting up...
CurvePlot.cpp:71: Setting Curve "Force X" Visible
```

COM1

ad defaultConfig for II: a

20110714152712_FT-Gui_log.txt

	Channel 0	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5
	3	3	3	3	3	3
	1	1	1	1	1	1

Sample Rate: 3500

Signal Damping: 100

inc: 100

acc: 5

Graph ✓ Graph ✓ Graph ✓ Graph ✓ Graph ✓ Graph ✓

Temperature 1: Graph

Temperature 2: Graph

Figure 8.3: Force/torque sensor GUI used to initialize, synchronize, and log the sensor data.

Robot Control Library

This library performs the robot arm control tasks. These tasks are divided into high level control (such as connecting and disconnecting to the robot arm hardware or to its model in the simulator and move the arm to its home position) and low level communication with the robot hardware or to the simulator to set and get joint values.

The library consists of two components: a motion planner and a control thread. The motion planner calculates the forward and inverse kinematics of the arm to generate the desired joint trajectories. The control thread processes the trajectory generated with the motion planner. The control thread is created and destructed by the control library and it is designed in a way to ensure a thread safe execution of the program.

Graphical User Interface (GUI)

The graphical user interface is designed to be as user friendly as possible when performing the experiments. It uses the robot control library to connect to either the simulation or the real system and then to control the arms.

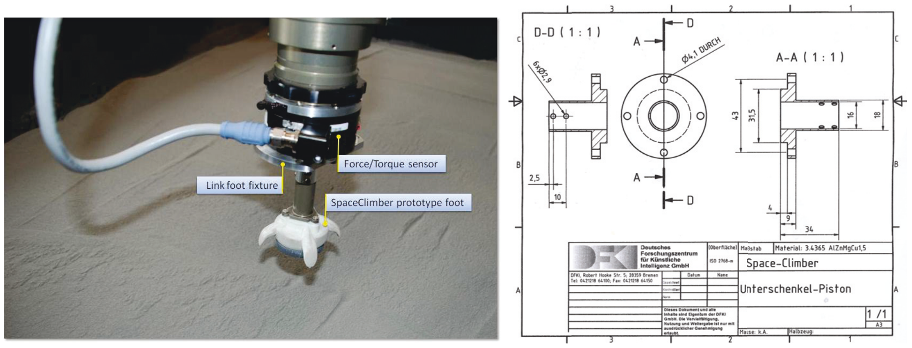


Figure 8.4: SpaceClimber foot connected to Mr.SemProm arm in the test bed (left) and details of the mechanical foot adapter to attach the foot to the arm (right).

The main functions of the GUI are monitoring the robot status (such as connection state and current joint angles), sensors handling and visualization, logging of experiment data, and keeping track of the user point list for the desired end–effector trajectories (in Cartesian or joint space).

The desired way points of the arm (with the attached test leg) are organized in a point list. This point list holds necessary information for the execution of that point through the robot control library such as point coordinates and execution time.

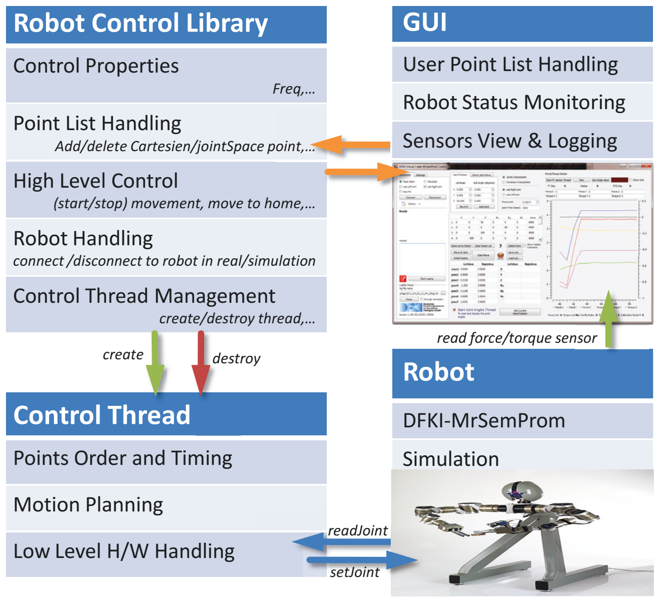


Figure 8.5: Schematic representation of the software components and control library.

The developed and implemented hardware drivers, robot control library, and software

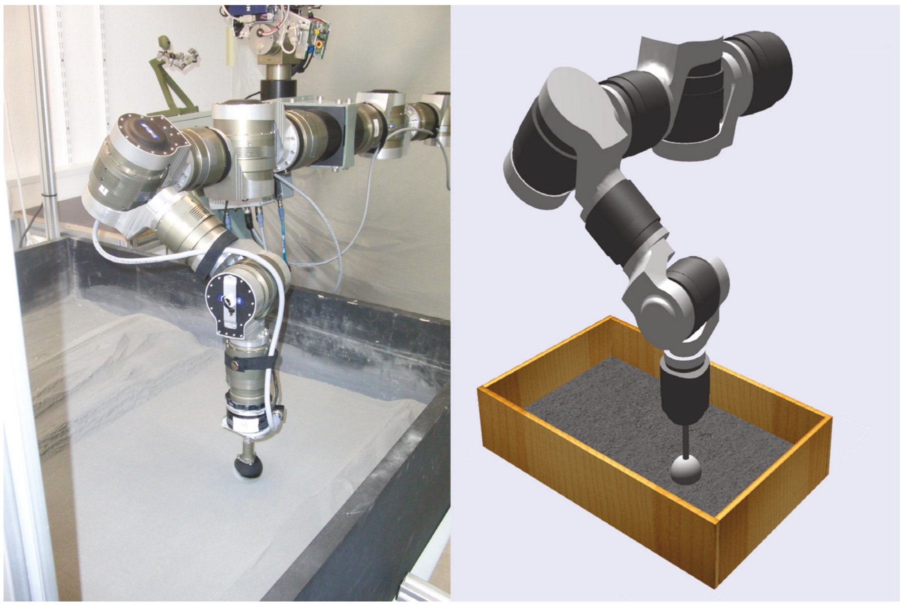


Figure 8.6: Real testbed and its simulation model.

GUI can be connected both to the real robot, as well as to a model of the robot in the simulation. Thus, the control of the robot arm and experiment scenario can be fully developed and tested in the simulation before the software is coupled to the real system.

8.1.4 Terramechanics Simulation Testbed

As a framework for the simulation, the extendable simulation software for physical systems (Verosim) [Ahmed et al., 2011, Yoo et al., 2010b] is used. This software is extended within the Virtual Crater project [Virtual Crater Project, 2012] with a new plug-in developed to model the soil-robot-interaction. Also a model of DFKI–MrSemProm arm is imported from CAD data and equipped with the appropriate joints and motors to closely resemble the real setup (see Figure 8.6).

The joint positions that are recorded during the experiments are then loaded into the simulation environment with a newly developed Verosim-plugin. These joint positions are used to control the model of the robotic arm so that its movement in simulation follows its real trajectory as in the real experiment. Data from a simulated six degrees of freedom force/torque sensor is also logged during simulation runs for comparison with the experiment data.

The simulator is used for visualization and testing of the test scenarios before experimenting on the real test setup. The objects created in the simulator can be controlled via a socket communication protocol from the graphical user interface (GUI) through the robot control library. It is also possible to use the simulator to generate the same movement of the foot in the real robot when walking from its simulation model and then supply this back to the GUI as a desired user point list.

8.2 Leg–Soil Terrain Interaction Experiments

In this section, a set of reference experiments is implemented and performed in the test environment (described in Section 8.1). The aims of these reference experiments are:

- to further investigate the foot–soil force behaviors,
- fit the performance of the PTP model to the reference experiments data through optimizing its corresponding parameters using optimization techniques, and
- to create a reference database of measurement data to be used for model verification and may be used later to determine the soil parameters.

In performing the designed reference experiments, the whole system including: different test objects, DFKI robot arm, and the designed control and logging software, is used.

In general, the reference experiments are defined with the following parameters:

- A_1: depth of moving into the soil with attack angle of α.
- A_2: distance moved through the soil.
- A_3: twisting in the soil with angle of β.

8.2.1 Experiment Results of Different Foot Shapes

The previously defined parameters (shown in Section 8.2.1) are used in combination in performing the experiments. The different test objects used in the experiments to contact the soil (shown in Figure 8.8) are described in Section 8.2.1.

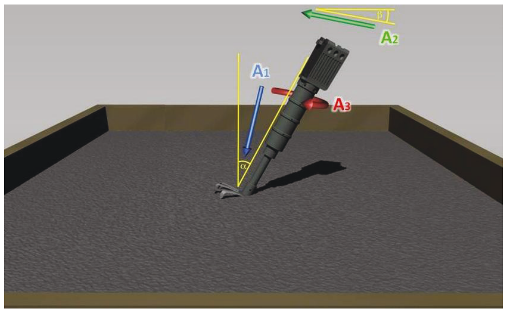


Figure 8.7: Reference experiment parameters.

Table 8.1: Description of the different test objects used in the foot–soil terramechanics interactions experiments.

Notation	Desription
C60	cylinder with diameter $\varnothing = 60$ mm, height $h = 105$ mm and mass of $m = 893$ g
B10	Cuboid with dimensions $L \times W \times H = 80 \times 85.44 \times 44$ [mm] and mass of $m = 900$ g
D70	DLR-Plate with diameter $\varnothing = 70$ mm and mass of $m = 204$ g
S60	SpaceClimber prototype foot with dimensions $L \times \varnothing = 100 \times 60$ [mm]
S60	SpaceClimber final foot with dimensions $L \times \varnothing = 112 \times 50.7$ [mm] and mass of $m = 221.1$ g

In all the experiments described next, the experiment configurations are given (see Section 8.2.1) and the resulting measurements including the arm seven joint angles (in [rad]), the end-effector tool-tip six force/torque senor (force in [N], and torque in [Nm]), and the arm end-effector Cartesian coordinates are automatically recorded and tagged with a time stamp through the designed logging software.

Table 8.2: foot–soil terramechanics interaction experiment configurations and parameters. T_c is the motion control cycle period and T_L is the measurements logging cycle period. Attack (α) and twist (β) angles are set to 0°.

Ref.	Object	A_1 [mm]	A_2 [mm]	A_3 [°]	T_c [ms]	T_L [ms]
RE44	C60	10	100	0	50	850
RE24	C60	20	100	0	50	850
RE09	D70	100	300	90°	50	850
RE30	D70	20	100	90°	50	850
RE54	D70	30	100	90°	50	850
RE20	D70	20	0	10°	50	850
RE37	D70	20	200	20°	50	50
RE45	D70	20	200	20°	30	50
RE41	B10	10	100	0	50	50
RE12	B10	10	0	20°	50	50

As an example of interpreting the entries of Section 8.2.1, the reference experiment RE41 was performed by commanding the arm to move the foot one step in an attack angle of 0° to the soil and to go into 10mm in depth into it. Then the foot was dragged in that depth for 100mm with a motion control cycle period (T_c) of 50 ms and the measurements were logged each (T_L) 50 ms. Figure 8.9 shows this experiment. Figure 8.10 presents a subset of the experiment results. In all these plots, the green vertical lines are the logging marker and the red ones are the motion way points markers (their labels are the point IDs which are also logged)

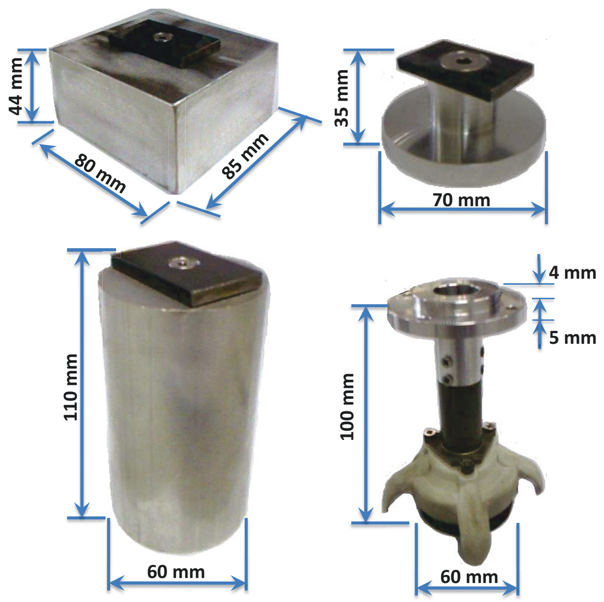


Figure 8.8: Some of the different test objects used in the terramechanics testbed reference experiments.

All the experiments defined in Section 8.2.1 are performed in the real test bed. It is concluded from the performance in the experiments that the setup can be used for further soil-foot interaction experiments.

8.2.2 Experiment Results Using SpaceClimber Foot

To create a reference database of real measurement data to be used in the robot-foot–soil interaction Terramechanics modeling and testing and for simulation verification, a set of reference experiments is implemented and performed in our soil testbed.

In these experiments, an exact SpaceClimber real foot (*S60*) is used to contact the soil. The foot shown in Figure 8.11 and details of the foot-adapter masses are given in Section 8.2.1. Each experiment is repeated for five times.

A three sets of experiments are designed. In the first experiment set, the robot arm is commanded to move the foot one step in an attack angle of 90° to the soil and to go from soil surface into 30 mm down then the foot is pulled up 30 mm. These actions are repeated for three successive times with a constant speed of 7.5 mm/s. Figure.Figure 8.12 show a sketch of this experiment.

In the second experiment set (sketched in Figure 8.13), the robot arm is commanded to move the foot one step in an attack angle of 90° to the soil and to go into 10 mm in depth into it. Then the foot is pulled up 10 mm, and this is repeated with increasing

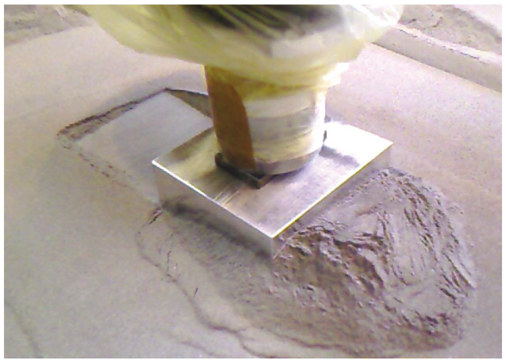


Figure 8.9: Reference experiment RE41 with cuboid–shaped test object (B10) used with the real Terramechanics Test Bed to test the soil interaction forces.

depth of 20 mm then 30 mm each with the same constant speed of 0.0075 m/s. Each movement is executed within four seconds, so the corresponding speeds are 2.5, 5.0, and 7.5 mm/s for the first, second and third step, respectively.

In the third experiment set, the robot arm is commanded to move the foot one step in an attack angle of 90° to the soil and to go into 20 mm in depth into it. Then the foot was dragged horizontally (in the x-direction) in that depth for 150 mm with a constant speed of 12.5 mm/s. Then the foot is pulled up 100 mm. Figure.Figure 8.14 show a sketch of this experiment.

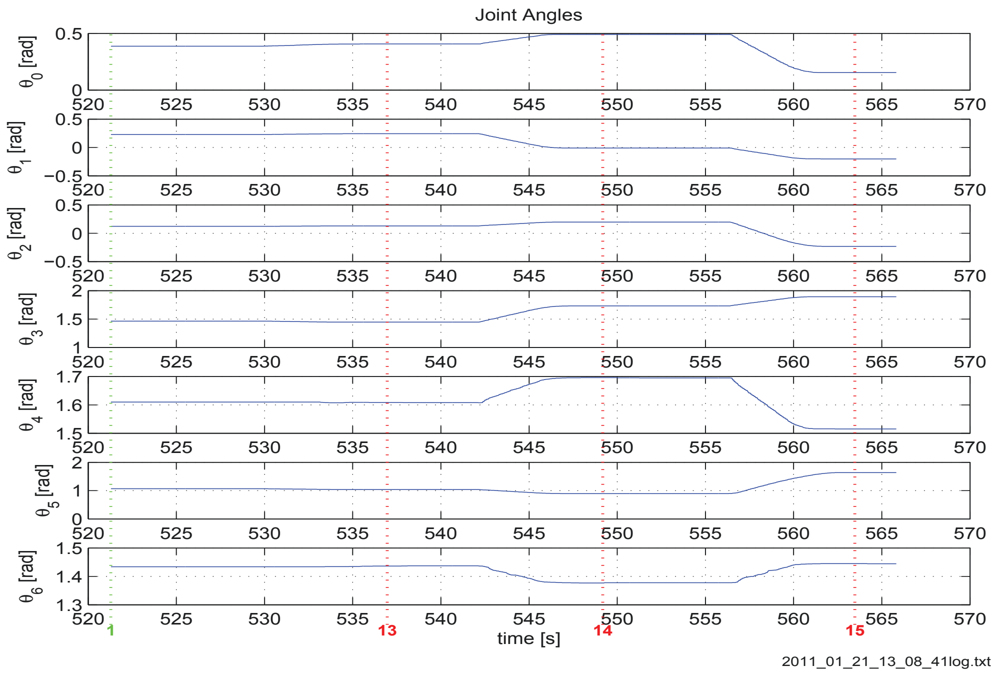
8.2.3 Results

In all the experiments described, the resulting measurements including the arm seven joint angles (in [rad]), the end-effector tool-tip six force/torque senor (force in [N], and torque in [Nm]), and the arm end-effector Cartesian coordinates are automatically recorded and tagged with a time stamp through the designed logging software.

The collected experiment data needed to build a database for analysis for the interaction between the robot legs and the ground and to study its characteristics. In addition, these data will be used next to test the proposed model test and verification of its performance. Figure 8.15 and Figure 8.16 show some of the results of the experiments performed.

8.2.4 Experiments with Vertical Movements

In the first set of experiments, a constant vertical force is applied to the foot. To fulfill this condition, the experiment control software is extended to perform active force control on the vertical force component of DFKI–MrSemProm arm (consequently, a constant force is applied to the foot). A passive compliance adapter attached between the force/torque sensor and the foot is constructed. The exact SpaceClimber real foot is used to contact



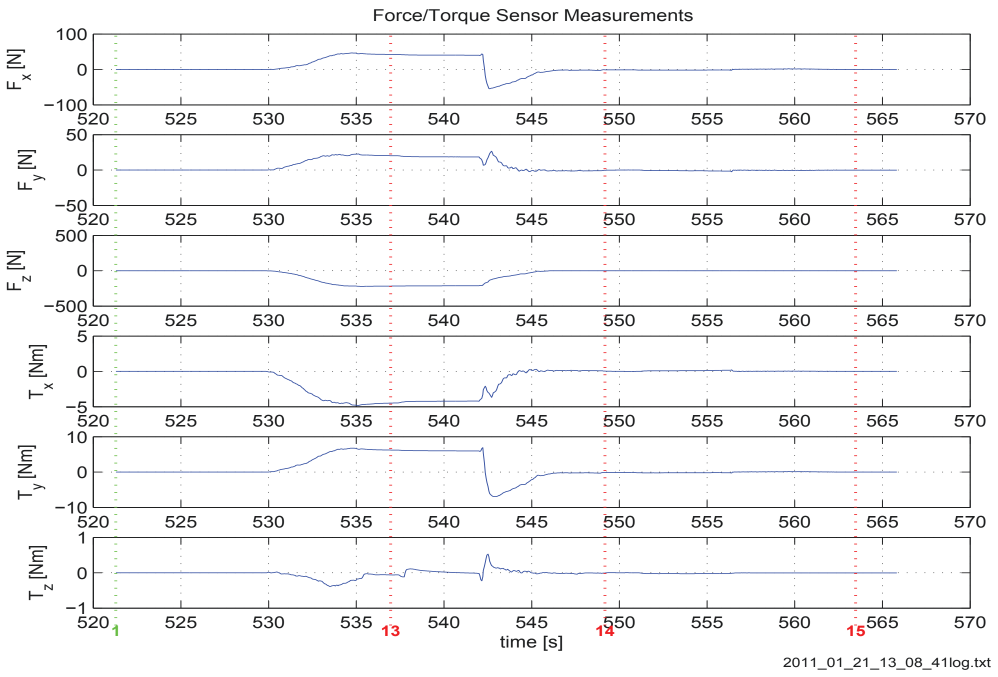


Figure 8.10: Joint angles and end–effector tool–tip force/torque results of the reference experiment RE41 using Cuboid (B10) to test the soil interaction forces.

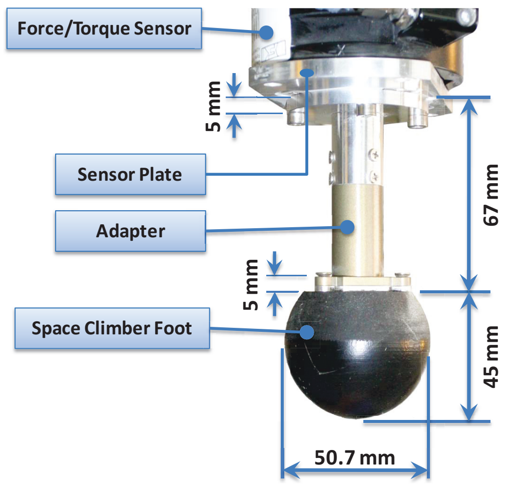


Figure 8.11: Detailed view of the the SpaceClimber foot assembly and the adapter connecting it to the force/torque sensor of the robot arm as used for the reference soil interaction experiments.

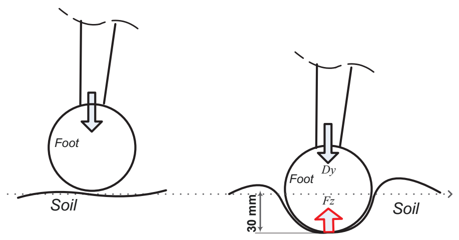


Figure 8.12: Sketch of the first soil contact experiment. In this experiment, the foot is moved from soil surface into 30 mm downwards into it and repeated for three times.

the soil. Figure 8.17-(left) shows the arm–foot assembly and figure Figure 8.17-(right) shows the details of the passive compliance adapter.

In these experiments, the foot was positioned perpendicular to the ground (i.e. the attack angle is 90°). Then, the force control is turned on with a nominal value of the desired constant force (F_d) of 50 N. The arm is moved with the leg into the soil. After about 5 seconds the arm is dragged laterally (in the x-direction) with a speed of 5 mm/s for 150 mm. During these movements, the force control maintains the normal force constant. Figure 8.18 show a sketch of this experiment.

In all experiments, the resulting measurements including the arm seven joint angles, end-effector tool-tip six force/torque senor, the arm end-effector Cartesian coordinates, and the foot internal sensor measurements (accelerations, IMU, four adjacent force sensors, and four adjacent pressure sensors) are automatically recorded and tagged with a time stamp through the designed logging software. Figure 8.19 shows the results of five of the experiments performed.

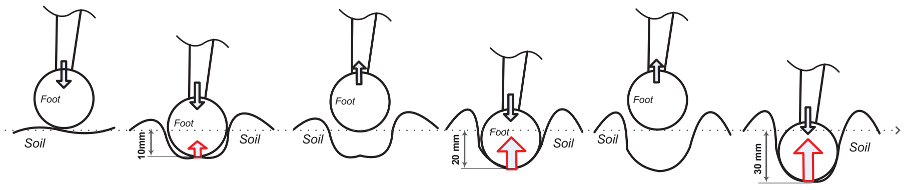


Figure 8.13: Sketch of the second soil contact experiment. The foot is moved for 10, 20, and then 30 mm into the soil while returning up to the soil surface between each step.

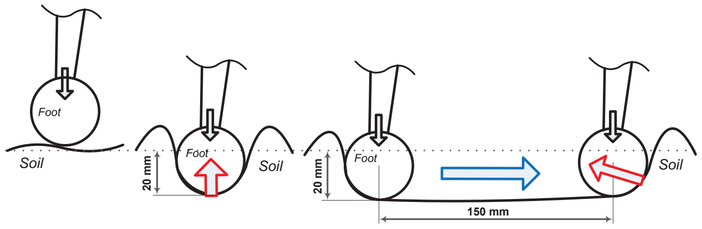


Figure 8.14: Sketch of the third soil contact experiment. In these experiments, the foot is moved for 20 mm into the soil then at that depth moved for 150 mm horizontally through the soil with a constant speed of 12.5 mm/s.

8.2.5 Experiments with Basalt Soil

In order to investigate the flexibility of the various implementations of the sandy soil mechanics has been replaced in the Terramechanics Testbed. The existing soil was replaced with basalt. This is the same material that is already in use at DFKI in a larger test facility in which the fully integrated CREX can be tested [Hall, 2014]. Thus it was possible the real data from experiments carried out with the entire SpaceClimber and not only with the individual components, to be compared with data from the simulation.

Experiments were carried out in Basalt (see Figure 8.20). Five exemplary records for a lateral experiment are shown in Figure 8.21. Further experiments are performed to verify the results and the model performance with a different soil type. to measure the normal interaction forces between the robot leg and the basalt soil type (see Figure 8.22).

The results from these experiments show a similar range of variation as the fine-grained sand. However, it is detected in various experiments that the speed of foot insertion into the soil has a much greater effect on the soil resistance. This effect could not be further experimented and studied with the existing test setup due to the lack of precision at higher speeds. Another difference is in the characteristics of the force curves that varies more significantly than the performance recorded with the finer-grained soil substrate. As illustrated in Figure 8.22, the response curves are close to each other within the first stage (up to ca. 25 mm of depth) while they are distributed to almost the entire range of variation after that region. We believe that the differences in the characteristics of the

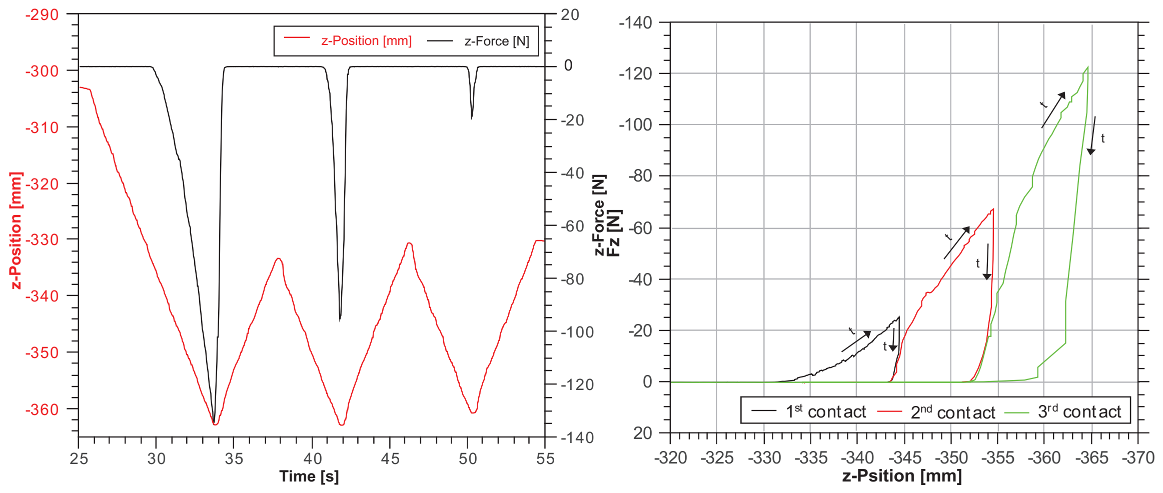


Figure 8.15: Exemplary results of the first and second sets of soil contact experiments (left and right, respectively). The vertical force is plotted against the foot penetration depth into the soil. The three contact processes with different depths are shown in different colors. The flatter each edge corresponds to the penetration into the ground while a continuously opposing force is built up through the soil. The steep edges correspond to puling the foot up and out of the soil. The time progress of the experiment is indicated by the t labeled arrows.

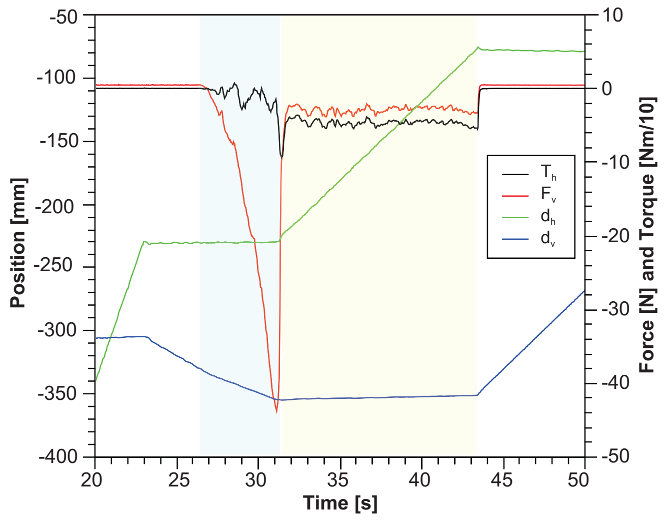


Figure 8.16: Exemplary results from the third soil contact experiment, blue shaded zone is for the soil contact (starting at $t \approx 26.5$ s) while yellow is for the lateral movement (from $t \approx 30.5$ s). T_h is the horizontal torque and the vertical force is (F_v). d_h, and d_v are the horizontal and vertical distances, respectively.

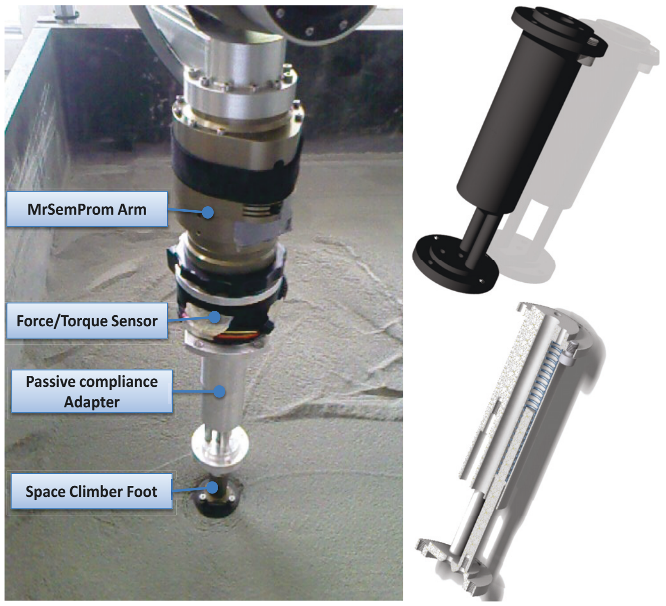


Figure 8.17: SpaceClimber foot attached to DFKI–MrSemProm arm through the passive compliance adapter (left) and the internal mechanical details of the passive compliance adapter (right).

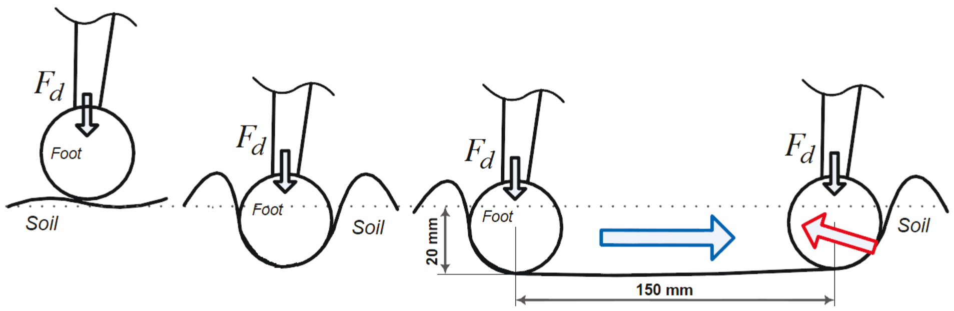


Figure 8.18: Sketch of the horizontal foot–soil force behavior experiment. F_d is the desired constant vertical force applied to the foot. The foot is dragged horizontally along 150 mm at a speed of 5 mm/s with F_d constant at 50 N.

force curves are due to changes in the soil parameters during the experiments as the soil becomes more compacted after the foot interacts with it.

8.2.6 Foot–Ground Contact Model Verification

The previously detailed experiments are organized mainly in two categories: experiments for measuring the foot–soil interaction force while moving the foot in the lateral direction.

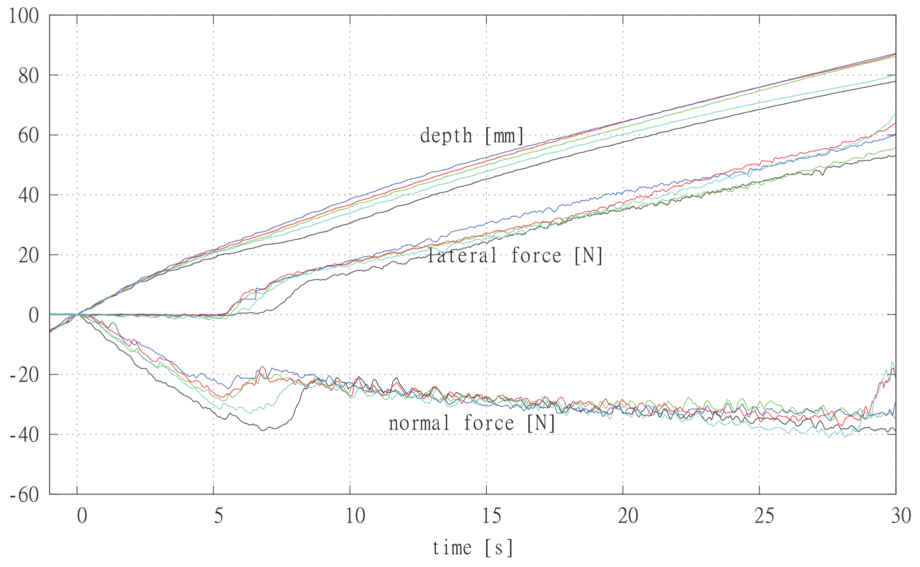


Figure 8.19: Results of the horizontal foot–soil force behavior experiments with force control.

The second category is that forces measured while moving in the vertical direction. Next, we are to use these data collected in the previous sets of reference experiments to test and verify the developed leg–soil contact model.

Lookup Table Generation

As presented in Section 4.3.1, the PTP Lookup table is generated from experiment data by first fitting these data to its theoretical model (as given in Equation 4.26) that interpolates the recorded foot–soil interaction force. Then the interpolating curve is quantized to the desired accuracy (as given in Equation 4.27).

Figure 8.23(left) presents this process for a vertical force experiment data to generate the lookup table values. The experiment was performed with a flat circular plate of 5 cm in diameter. The value fit function and value quantization are also displayed.

To test the generated model, the same experiment is performed in simulation. The same motion commands are applied to a spherical SpaceClimber foot. The comparison between the PTP model performance and the real expirement is shown in Figure 8.23. As clearly seen from performance comparison, the model performance closely follows the performance of the real experiment. Hence, it can be concluded that the model can be utilized to predict the contact forces resulting from foot–ground contact.

Further set of experiments were done with a simple foot shaped as a cuboid dimensions of $2 \times 3 \times 5$ cm^3 (shown in Figure 8.10). To measure the contact forces, the foot was pressed into the soil and then pulled horizontally through it. The use of the simple

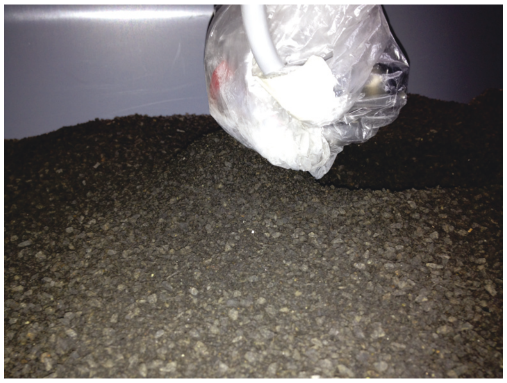


Figure 8.20: Lateral Soil Mechanics reference experiment with basalt sand.

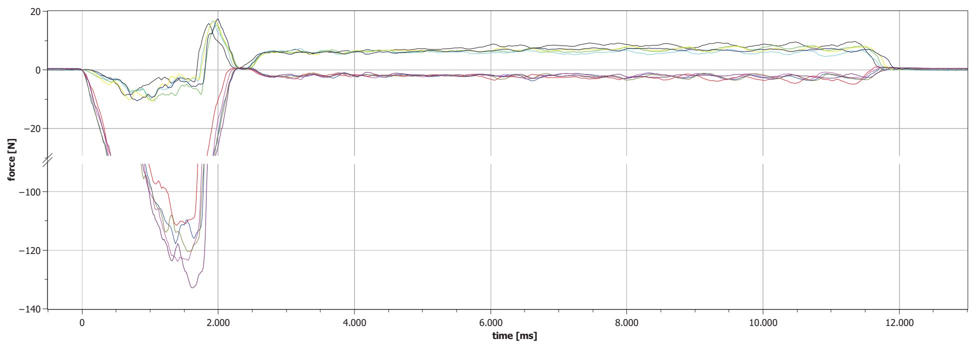


Figure 8.21: The results from five of the experiments done for the lateral forces with Basalt soil

shaped leg of a cuboid in this experiment has the advantage that the surface of the cube pressed against the soil is known. Consequently, the soil reaction force can be accurately converted into pressure which is needed in the contact model. Figure 8.24 presents data collected in one of the experiments. As evident from the response plot, the lateral force seems to be constant over long distances (the small variations are due to measurement noise). In this case, the lookup table is trivial as it will be designed to deliver a constant value.

For further verification of the PTP model, the robot spherical foot was pulled through the soil in the simulation and the results are compared with the reference experimental data. This comparison is shown in Figure 8.25. It is worth noting that in Figure 8.25, the decrease in the normal force observed when applying the lateral movement (at $t \approx 8$ s) in the real experiment force response is qualitatively reproduced by the model response. However, the simulation response has a kind of saturation in both the normal force and the lateral force that does not occur in the real experiment. Also, the noise in the lateral force should be reduced. The observed difference in the constant value of the lateral

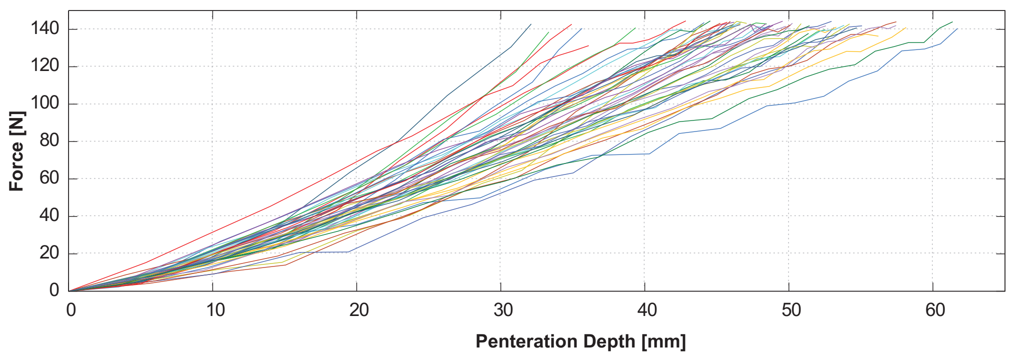


Figure 8.22: Results from all 48 experiments conducted to measure the normal forces of the foot–soil interaction with the basalt soil type. The main characteristic of the normal forces of the foot–soil interaction measured data with the basalt soil type is that the force response curves within the first stage of contact (up to ca. 25 mm) are relatively close to each other while they are afterwards distributed (after ca. 25 mm) on the whole range of variation.

force is due to the unmodeled bulldozing effect.

Another comparison between the real reference experiment and simulation with the PTP approach is shown in Figure 8.26.

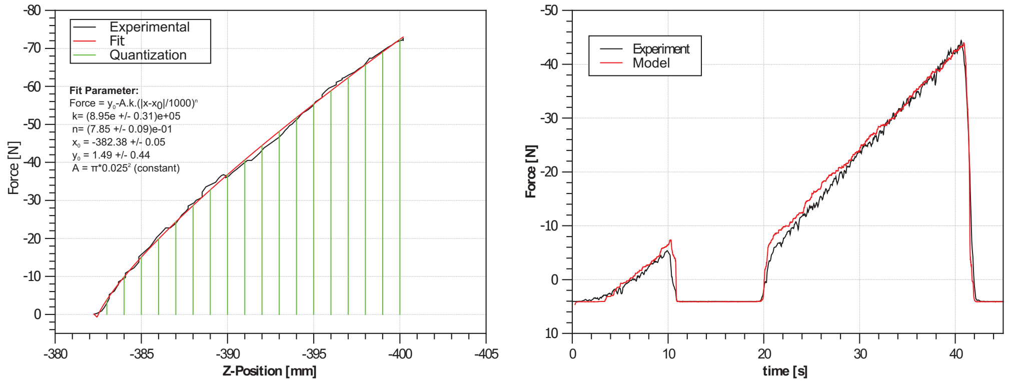


Figure 8.23: Left: results of a vertical force experiment with a circular plate ($\varnothing = 5$ cm) (black), a value fit function (red), and a value quantization (green). Right: the same experiment repeated with SpaceClimber spherical foot (black) and model results of the same experiment based on the lookup table of values from the first experiment (red).

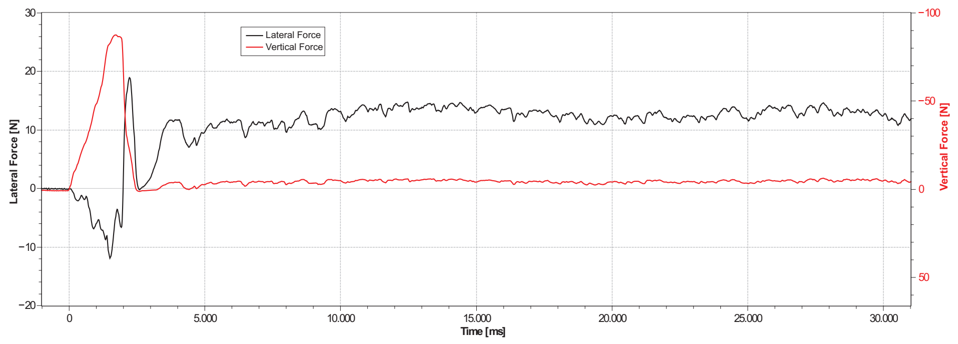


Figure 8.24: Data collected from one of the soil contact reference experiments. The experiment was performed with a cuboid of $2 \times 3 \times 5$ cm^3. In the lateral movement, the 3×5 cm^2 face was pressed against the soil.

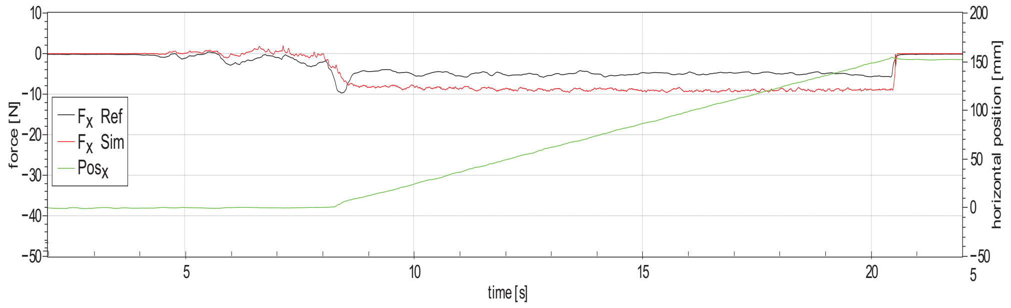


Figure 8.25: Comparison of real experiment results with simulation of the PTP model for lateral movement. The robot spherical foot is used in these experiments. In the plots, F_x is the lateral contact force from real (black) and simulation (red) experiments. POS_x is the lateral displacement (green) of the foot into the soil.

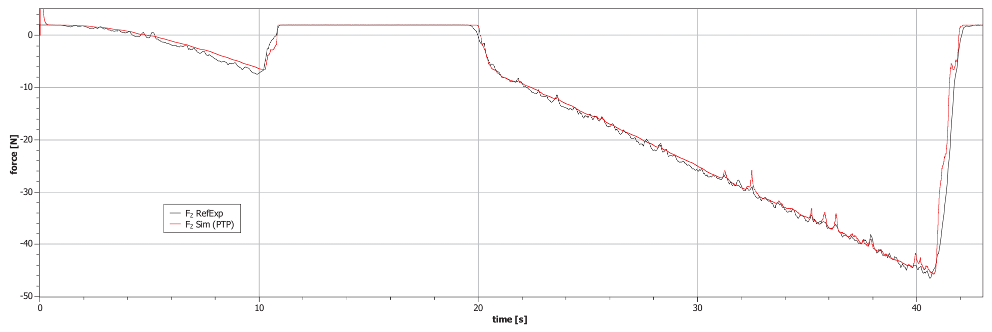


Figure 8.26: Comparison of the measured normal forces from the real experiment and the virtual experiment with PTP approach.

8.3 Short-Range Terrain Classification Experiments

Field experiments were performed on the "[DFKI Robot Test Track, 2015]". It is an outdoor testing facility DFKI–RIC with 15 challenging obstacles and an area of more than 500 m^2. The obstacles can be used for performance evaluation of walking, driving and climbing robots. The test environment has different terrain types such as sand, rocks, grass, and gravel (see Figure 8.27).

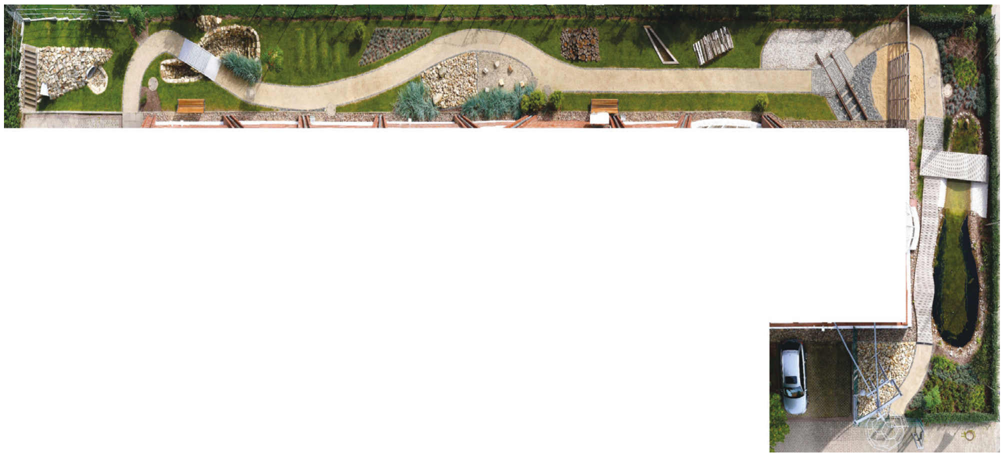


Figure 8.27: DFKI-RIC Robot Test Track.

The six legged robot Crater Explorer (CREX) was used (see Figure 8.28). The robot was set to walk with a constant speed on three different terrain types: gravel, grass, and tile.

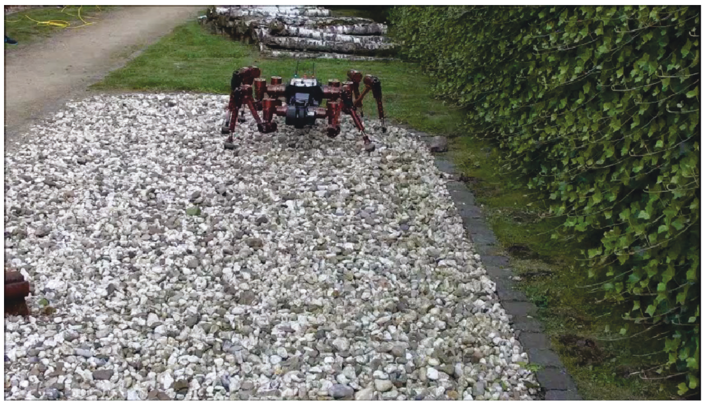


Figure 8.28: CREX robot on the test track.

8.3.1 Experiment Data Preparation

Approximately 600 sensor and internal measurement values (channels) were logged by the robot hardware. Figure 8.29 shows a sample of the raw values of the logged sensor values.

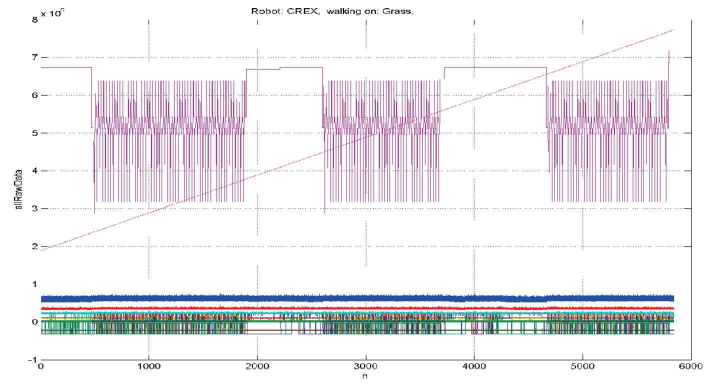


Figure 8.29: Raw sensor values logged by the robot hardware.

These experiment data are stored in log files with the comma separated values (.csv) format. The specification of the log files containing the data to be processed are prepared in YAML files. The data files are then loaded by pySPACE into a database and various machine learning operations are performed.

Figure 8.30 shows the node chain specifications that is used to process the robot proprioceptive senor data. It illustrates the concatenation of different node categories (introduced in Section 5.3. Data samples for this node chain consist of multiple channels (sensors) and multiple time points, so that after loading, we obtain windowed time series. Each data sample is then processed as specified: each channel is standardized, reduced in sampling rate and lowpass filtered. Then, the data are equally split into training and testing data to train the supervised learning algorithm (the LibSVM Support Vector Machine). Included in this node chain is a hyper-parameter optimization (grid search) of the complexity parameter of the classifier. This is done with five-fold cross-validation on the training data. Finally, performance metrics are calculated respectively for training and testing data.

8.3.2 Node Chain Description

The node chain used to process the experiment data is descried as:

1. **Stream2TimeSeriesSourceNode**: Transformation of streaming data to windowed time series. This node contains an interfaces the streaming dataset to provide it with a windowing specification and then get a data generator.

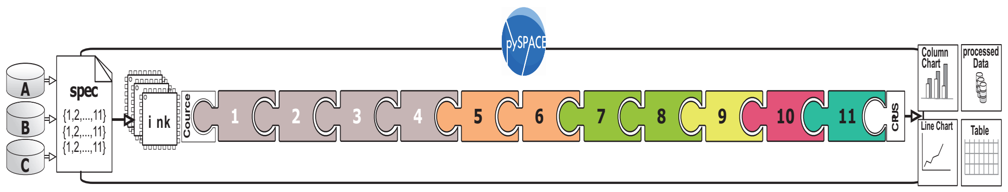


Figure 8.30: Processing flow of pySPACE node chain operations for the robot terrain data. A, B and C are the datasets of the experiment data, which is processed as specified in the spec file. The processing is then performed automatically. As a result, it produces performance metrics. In addition it can produce new data and also visualizations and performance charts. The puzzle symbols illustrate different modular nodes, e.g., a node to transform streaming data to windowed time series (1), a cross-validation splitter (5), and classifier (7). They are concatenated to a node chain. The detailed description of all nodes used is given in Section 8.3.2.

2. **NaN2Number**: Converts all the NaN enetries in the data set to 0.0.
3. **Int2Float**: Converts all the entries in the data set to either a double or longdouble precision.
4. **ChannelNameSelector**: Project onto a subset of channels based on their name. This node reduces the number of channels of the signal by projecting onto a subset of channels that are selected explicitly by the user via their name.
5. **Decimation**: Downsampling with a preceding FIR filter. The decimation is performed by doing a downsampling with a preceding filtering by a corresponding low pass filter beforehand. According to Shannon-Nyquist's sampling theorem, only frequencies below $\frac{1}{2}target_{frequency}$ should be retained. The decimation is applied in multiple steps, if the sampling factor is too big. The filtering procedure is applied by an FIR filter, and only for values that are significant due to the downsampling procedure.
6. **Statistical_Features**: This node computes statistical features like raw and central moments of k-th order, median, min etc. The raw moment of k^{th} order is ($k = 1$ is equivelent to the mean):
$$\frac{1}{n} \cdot \sum_{i=0}^{n} (x_i^k) \tag{8.1}$$
7. **CrossValidationSplitterNode**: During benchmarking, n pairs of training and test data are generated, where n is configurable via the parameter splits. The n test datasets are pairwise disjunct. Internally, the available data is partitioned into n pairwise disjunct sets $s_1, ..., s_n$ of equal size (the "splits"). The i-th pair of training and test data is generated by using s_i as test data and the union of the remaining datasets as training data.

The partitioning is stratified per default, i.e. the splits have the same class ratio as the overall dataset. Per default, the partitioning is based on shuffling the data randomly. In this case, the partitioning of the data into $s_1,...,s_n$ is determined solely based on the run number (used as random seed), yielding the same split for the same run_number and different ones for two different run_numbers.

8. **GaussianFeatureNormalization**: Transform the features, such that they have zero mean and variance one. A class that normalizes each dimension of the feature vector so that it has zero mean and variance one. The relevant values are learned from the training set.
9. **MultiClassLayerNode**: Wrap the one vs. rest or one vs. one scheme around the given node. The given class labels are forwarded to the internal nodes. During training, data is relabeled. Everything else is the same as in the base node.
10. **PerformanceSinkNode**: Calculate performance measures from standard prediction vectors and store them. It takes all classification vectors that are passed on to it from a continuous classifier, calculates the performance measures and stores them. The results can be later on collected and merged into one tabular. Essentially, the weighted accuracy is calculated by

$$\text{Weighted\_accuracy} = weight \cdot TPR + (1 - weight) \cdot TNR \tag{8.2}$$

where TPR is the True-Positive-Rate, and TNR is True-Negative-Rate.

8.3.3 Results

The classification results of the tested three terrain types are given in Figure 8.31(upper, left). The classifier gave very satisfactory results as the mean overall accuracy was above over 93%.

From the huge number of raw input channels (approximately 600 sensor), the machine learning framework pySPACE selected only 72 channels of them which affect the classification. The effect of number of channels removed per each step and the length of the time window samples are shown in Figure 8.31(down). It is noted that for higher classification accuracy, we should have a longer window or more channels. This has a direct effect on the training and classification time as given in Figure 8.31(upper, right).

The plot in Figure 8.31(upper, right) indicates that when we remove a larger number of channels, we will need a longer time for training and classification. This seems to be logical as the information content relating to the terrain interaction characteristics in the channels decreases with fewer number of them which in turn requires longer time of the classifier to learn the difference between the terrain classes. we different time window sizes and a window size of 2.5 s gave the best results.

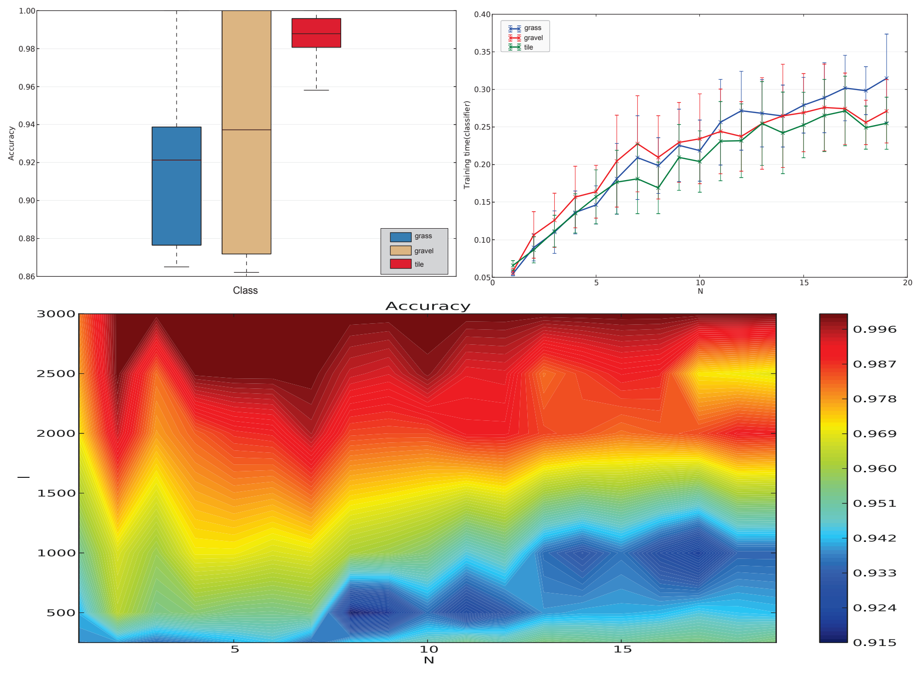


Figure 8.31: Results of the short-range terrain classification experiments. In the plots, N and l are the number of channels to be removed from the experiment data and the number of time samples per window, respectively.

8.4 Visual Classification Experiments

Here we describe the setup and Experiments in which we conducted our visual terrain classification. The experiments were performed in an offline supervised classification fashion.

8.4.1 Image Datasets

Since there is no available terrain image benchmark (to the best of our knowledge), to benchmark our classification algorithm, we created a large dataset of outdoor terrain images with six different terrain types: Tile, pavement, grass, sand, and gravel. Each terrain class has at least 80 samples captured on different places. These sample images, having a resolution of 4320×3240 pixels, were taken with the camera under different lighting, sun angle and weather conditions. These varing conditions make this dataset challenging to classify with simply color or texture alone. Figure 8.32 shows four different images from each class to give an impression of differences between samples.

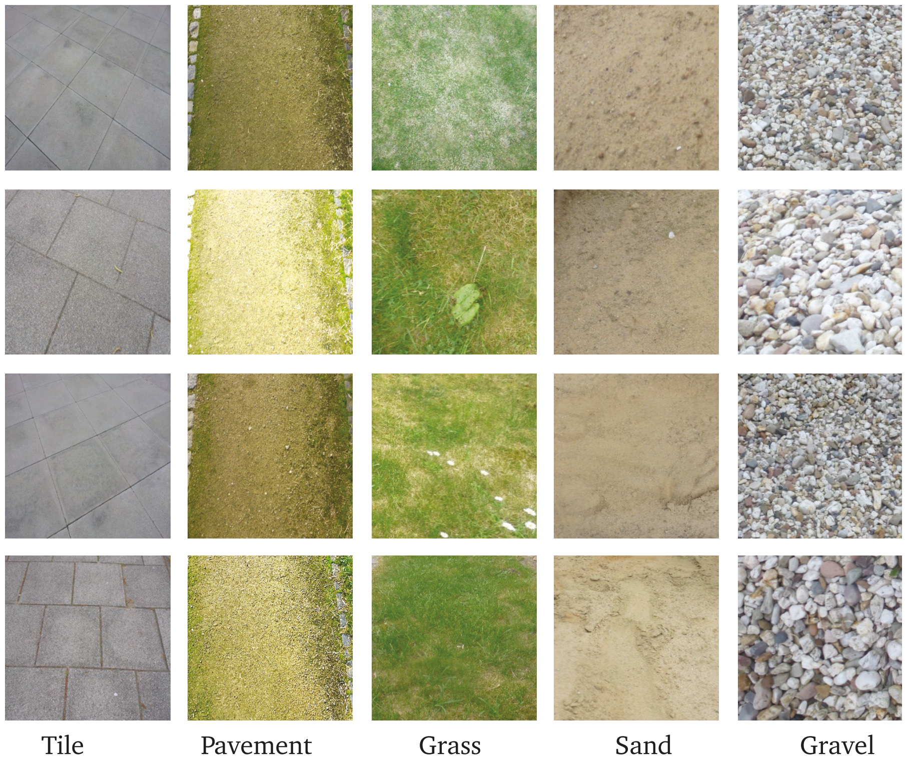


Figure 8.32: Sample images of different terrain classes under different image conditions.

To create a suitably large quantity of entries for training and testing, each raw camera image was divided into square sub-images of specific image dimensions. Each sub-image only one terrain type. This results in a dataset of 702 images split into three sets with 234 images each. Third of the images in a set are used for training and the two thirds are used for verification of the classifier. The images are randomly selected and assigned to the set.

8.4.2 Classifier Parameter Selection

Our long range visual terrain classifier has a number of parameters. In this section, we present the experiments done to investigate their effects on the classifier and to tune these parameters for optimal performance.

Table 8.3: List of parameter tuning experiments for the long–range visual terrain classifier. In each experiment the range is given in the form (`start:step:stop`) which means the parameter initial value is set to `start` and its last value is `stop` inbetween the value is increased by `step`.

Parameter	Range
number of vocabulary words	10:10:400
number of k-mean iterations	5: 5 :200
number of features per image	20:10:300
image size	192:64:576
number of training samples	1: 1 :40

Selecting Number of Training Samples

We applied both feature and color-based classifiers to our terrain image set. Figure 8.33 depicts the observed accuracy of each classifier as the number of training samples increases. The accuracy values of the SURF feature descriptors are quite always higher than the color-based. The color-based classifier exhibits less accurate performance due to color differences between samples. Both methods give nearly constant accuracy after a specific number of training samples. More over, the SURF method reaches over 90% of accuracy for the five classes.

The results suggests taking more than five samples for SURF training as the classification accuracy is almost constant for higher number of samples.

Selecting Number of Vocabulary Words

The selected classification technique is a feature-based one that uses the SURF extracted kypoints. This fact directly implies that the number of visual words in the vocabulary has a direct correlation with the classification accuracy. Figure 8.34 illustrates that few

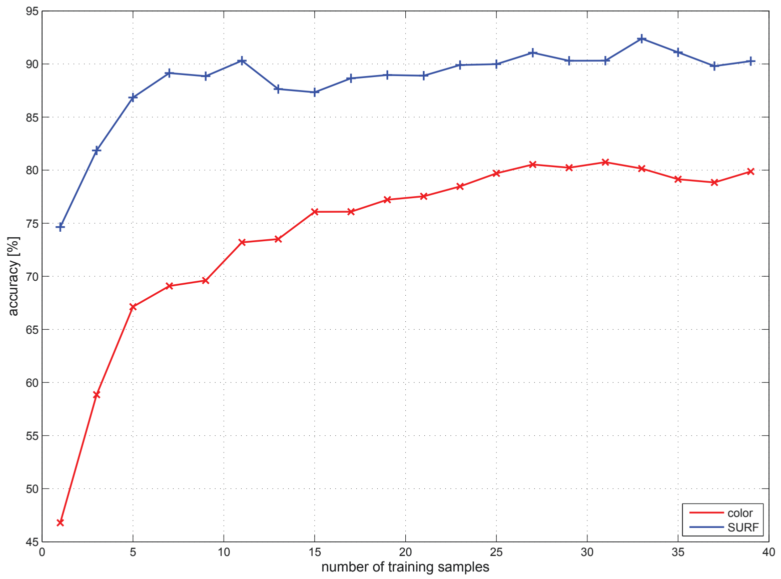


Figure 8.33: Accuracy of the SURF and color-based classifiers with the terrain image set.

words, such as 10 or 20, would not allow for the classifier to accurately represent each terrain type. On the other hand, it also indicates that too many words may not necessarily help improve the classification accuracy. Our classifier accuracy generally improved with word count up to about 150 words.

Selecting Number of k–`means` Iterations

This experiment is performed to find the necessary number of k–`means` iterations for optimal performance. Figure 8.35 shows that the number of clustering iterations has little effect on the accuracy. This is due to the k–`means` clustering algorithm that always converge in a finite iterations, so after that more iterations will not be beneficial. In addition, the k–`means++` probabilistic initialization procedure is a very effective method for our classification problem.

Selecting Number of Features

As the number of features has a major effect on the classification process. We can expect that more features means better accuracy. On the other hand, too much of features will flood the classifier which will increase the computational load and the storage requirements hence decreasing the overall performance. A tradeoff between accuracy and performance is found to be at 250 features per image as shown in Figure 8.36.

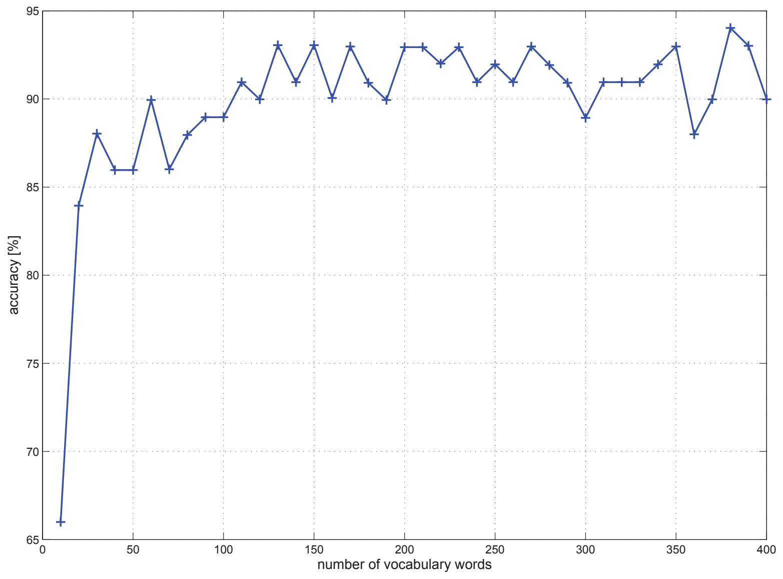


Figure 8.34: Accuracy terrain classifier with the number of vocabulary words. The accuracy generally improved and keep almost constant with word count up to about 150 words.

Selecting Image Size

As other parameters, we need choose the optimal image size to properly classify with the minimum image size possible. In this experiment, The terrain image is clipped to be of square dimensions starting from 192×192 up to 576×576 pixels in steps of 64 pixels.

The results of this experiment is presented in Figure 8.37. It is observed that an image size of 448×448 pixels gives the best performance. This experiment is performed also with the color-based classifier. The results again emphasize the inferior performance of this classifier in comparison with our classifier even with different image sizes

8.4.3 Visual Classification Results

The parameter values optimized in the aforementioned experiments are used with our visual terrain classifier to learn and classify the datasets for terrain type identification. The classification results are given in Table 8.3. The same data is plotted in Figure 8.38 for visualization.

We can observe that our classification method has 98.8% classification accuracy in training phase and 94.6% in testing phase and it thus the classifier has an average of 96.7% accuracy. Here it is clear that the method performs best with all terrain types except pavement and sand. This is due to the too much similarity between the two classes (visually and most of the time in features also). This makes it difficult, even for a human, to distinguish between them.

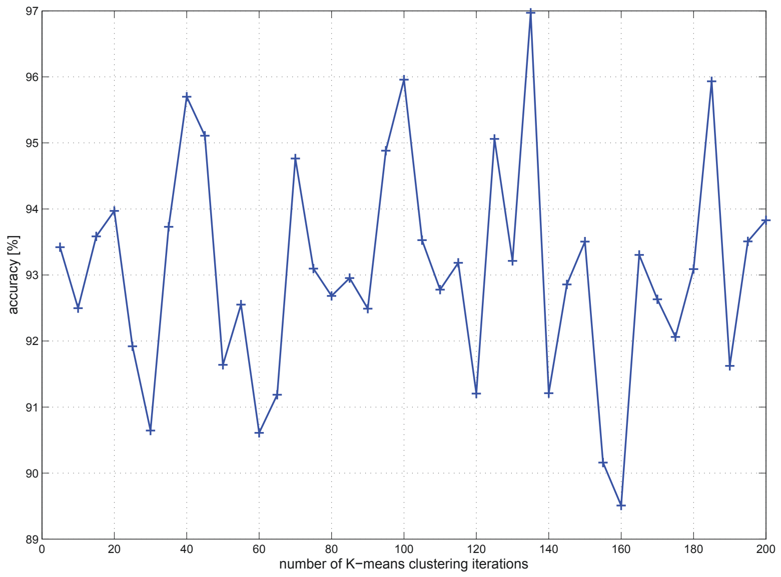


Figure 8.35: Accuracy of the SURF and color-based classifiers with the terrain image set.

Table 8.4: Long–range visual terrain classification results for the five terrain classes with homogeneous terrain images.

	Training			**Testing**		
Class	Set 1	Set 2	Set 3	Set 1	Set 2	Set 3
Tile	95.8	100	100	100	100	100
Pavement	94.4	100	100	83.3	94.4	88.9
Grass	100	100	100	100	91.7	100
Sand	94.4	100	100	88.9	88.9	94.4
Gravel	97.2	100	100	94.4	100	94.4

8.4.4 Heterogeneous Terrain Image Classification

While there is no straight forward performance metric for heterogeneous terrain classification, we adopted the method of hand labeling the images to create ground-truth images. Generation involves labeling the whole images with colors. A color coding is chosen such that each terrain type is assigned an RGB color which represents this terrain type. Then all pixels containing that terrain type in each image are manually painted with that color. All of the images are thus color coded. All of the areas which do not represent any terrain are neither painted nor removed. Figure 8.39 shows an image with multiple terrain types and the corresponding ground-truth image.

The image is then goes into the classification process for the terrains to be identified. The algorithm for ground-truth verification goes through all of the classified images

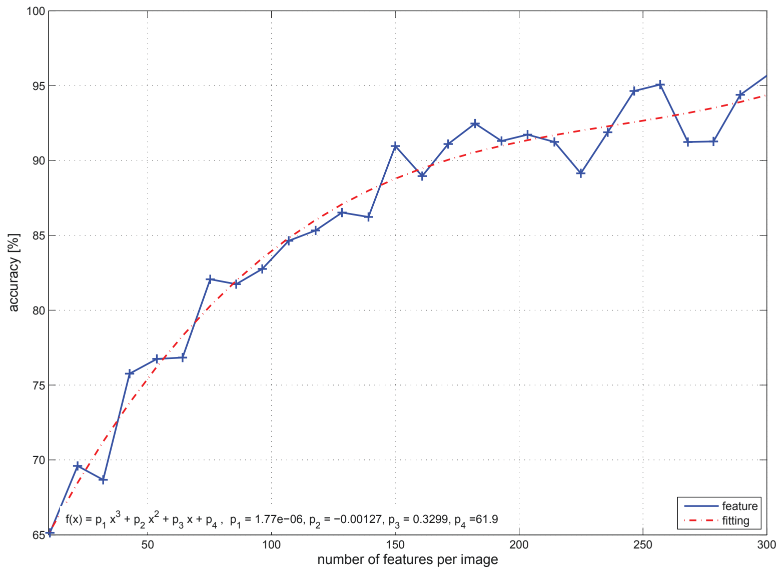


Figure 8.36: Accuracy of the SURF and color-based classifiers with the terrain image set.

and the corresponding ground-truth images. For each pixel of the image, it looks at the corresponding pixel in the ground-truth image and checks for the color code representing the terrain type and the precision value of the classifier over the entire dataset was obtained. Precision is the fraction of retrieved instances that are relevant. High precision means that an algorithm returned more relevant results than irrelevant. It is calculated as:

$$Precision = \frac{T^+}{T^+ + F^+} \tag{8.3}$$

where T^+ are true positives (actual positives which are correctly identified as such) and F^+ are false positives.

Table 8.5 shows the final confusion matrix calculated with our classification method. Here, 30 samples of images containing multiple terrain classes are used for evaluation. Although the method shows good classification, it has less true positive rate for sand but it is a more robust against false classifications.

Our method generated visually intuitive results as shown in Figure 8.40. The algorithm consistently identified homogeneous patches for all classes. In some cases, boundaries led to misclassification, as can be noticed in the last two images in the figure. Overall, the method classified images on a fine resolution and generated fairly good results.

8.4.5 Computational Complexity

Not only the accuracy of a classifier but also the consumed time for the classification task is an important value to be considered. Despite of its limited accuracy, the color-based

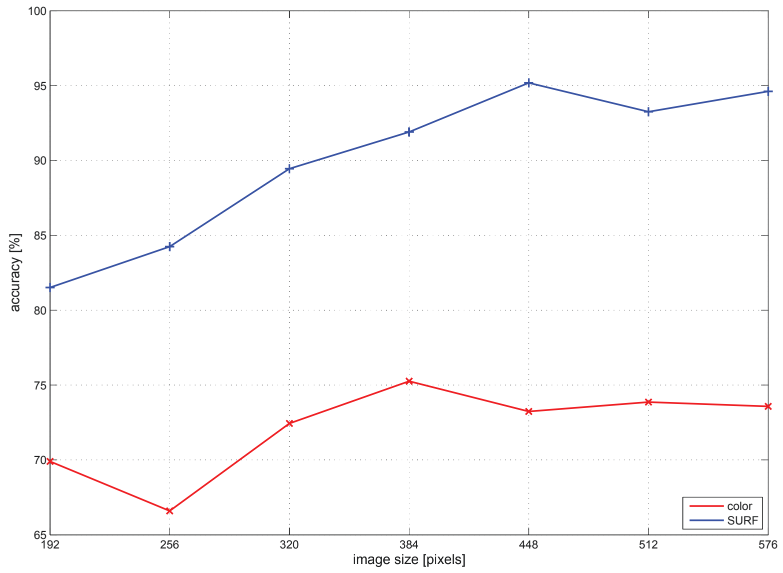


Figure 8.37: Results of the experiments to select optimal image dimension. The image is a square of size starting from 192 × 192 pixels.

Table 8.5: Long–range visual terrain classification results with images containing multiple terrain types. The results are represented by the confusion matrix for heterogeneous terrain images.

Class	Tile	Pavement	Grass	Sand	Gravel
Tile	100	0	0	0	0
Pavement	0	88.9	0	11.1	0
Grass	0	0	100	0	0
Sand	0	11.1	0	88.9	0
Gravel	0	0	0	0	100

classification is a computationally cheap method. It yields frame rates up to 30 fps (independent of the image content but limited by the camera) and has a fast training phase which takes at most 10 seconds for the terrain image set.

Our method requires more computational power due to the feature extraction step. In the worst case the classification lasts 0.2 seconds and in the best case 0.05 seconds with the hardware described in Section 3.3. Better hardware provides up to 5 fps during the classification (depending on the number of features detected in the image).

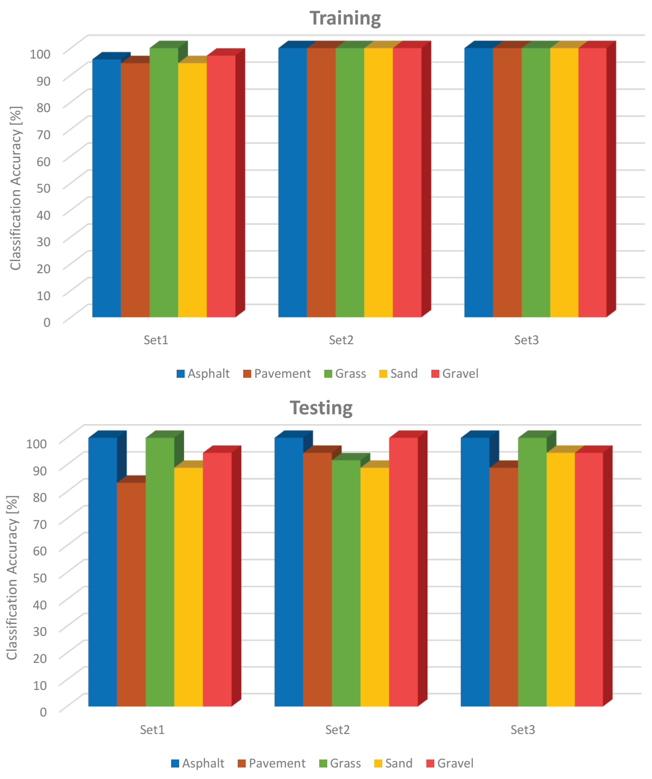


Figure 8.38: Long–range visual classification results of the five terrain classes in training (top) and testing (bottom).

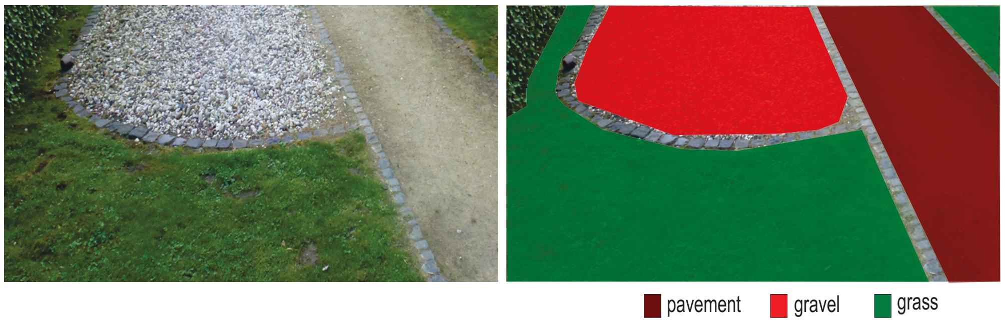


Figure 8.39: Camera image containing multiple terrain types (left) and its color coded hand labeled ground-truth image (right).

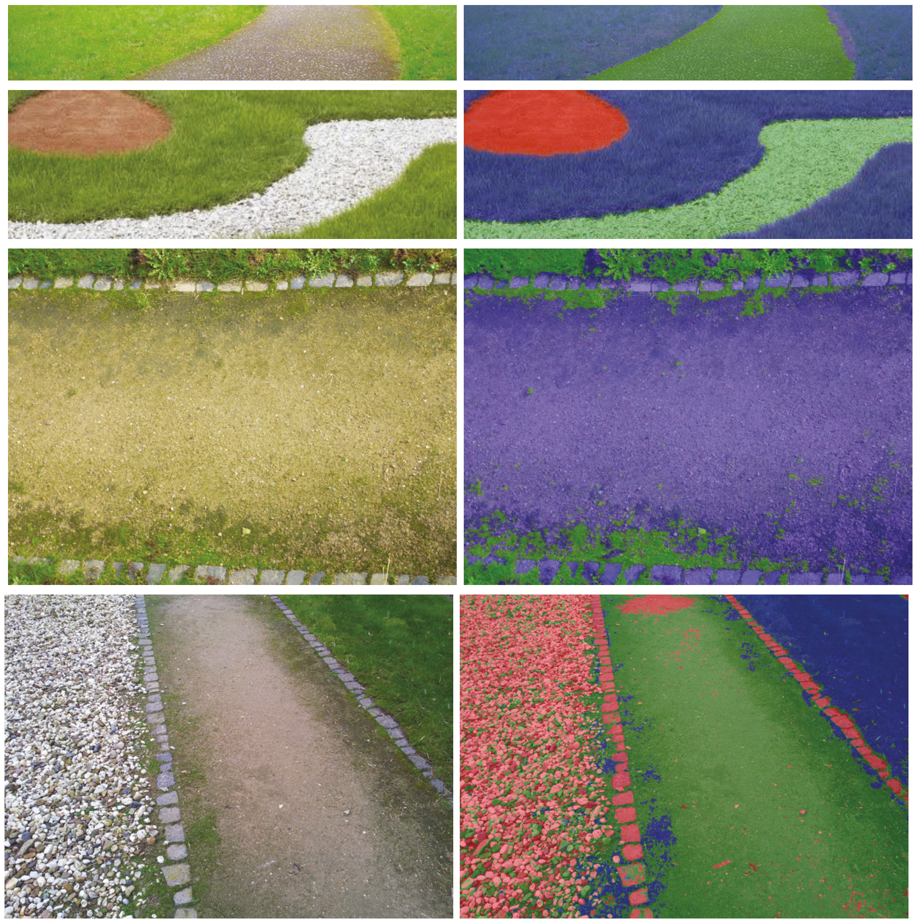


Figure 8.40: Accuracy of the SURF and color-based classifiers with the terrain image set.

8.5 Path Following Experiments

The designed controller is first tested as a stand-alone library encapsulated into a MATLAB/Simulink S-function (see Figure 8.41). For a precise simulation and to be as close as possible to reality, the implemented controller is interfaced to a detailed and dynamic robot model in Adams[1] within a cosimulation framework [Ahmed et al., 2010b] that also connects to the robot MONSTER mid-level controller for walk pattern generation. The model in Adams takes mechanical constraints and dynamic effects into consideration.

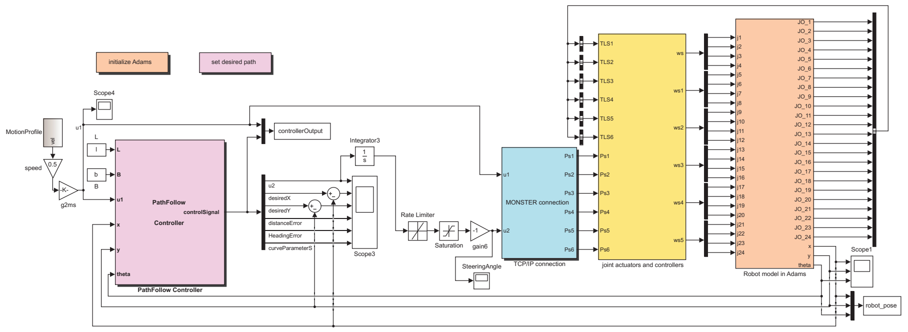


Figure 8.41: Path following controller implementation in Simulink model within a cosimulation framework that also connects to the robot MONSTER mid-level controller for walk pattern generation.

First tests were done with the control library and the robot model in the cosimulation to test its functionality. The simulation results are given in Figure 8.42.

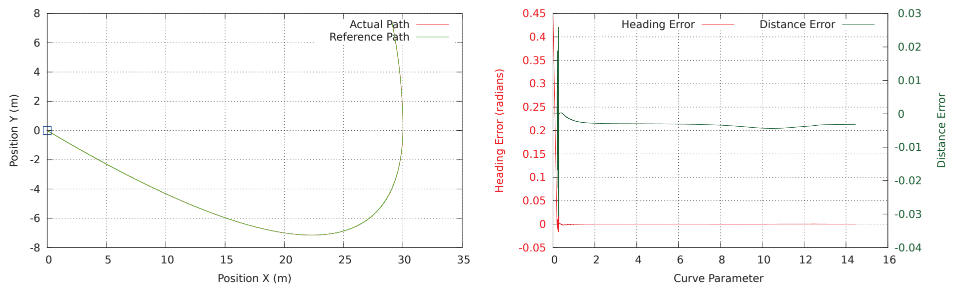


Figure 8.42: Path following controller simulation results.

Then, the control library is encapsulated into a ROCK component and integrated into the robot to experiment with it and to optimize its functionality.A desired path was extracted from a field experiment performed with the real robot. In this experiment the robot was commanded to move in an arbitrary path and its GPS coordinate points in addition to

[1] `http://www.mscsoftware.com/product/adams`

other sensor values were logged. These coordinates were supplied to the controller to move the robot in simulation. The results are given in Figure 8.43.

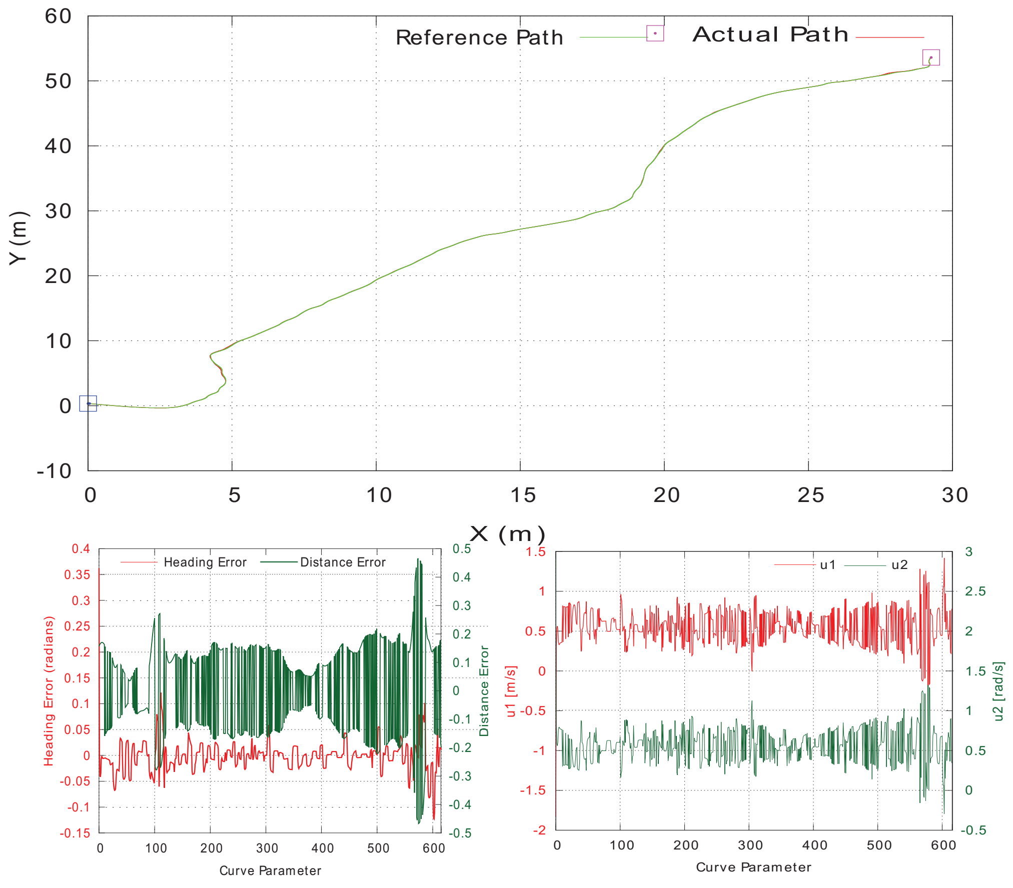


Figure 8.43: Path following controller simulation results for an arbitrary desired path

From these results, it can be seen that the system converges to the reference trajectory asymptotically. Once the robot converges, it follows the trajectory very closely. The convergence can also be seen in the error graph, where the initial conditions are also visible. It is clear from these figures that the controller does not attain extremely large values, and is bounded. This is an essential property for real systems.

8.6 Camera Calibration Experiments

In this section we describe the experiments performed to obtain the specific calibration information of the used camera, which are essential for the transformation from image coordinates to world coordinates that is a fundamental step needed for the projection of the classified image patches to the world map used in the path planning step. These parameters include geometry (e.g. image width and height), intrinsic, and extrinsic parameters.

The camera to be calibrated is the one used on the CREX robot head. It is a Prosilica GC1380CH with Sony ICX285 ExView CCD. The image resolution is 1360×1024 at 30 fps (max). The model plane used for calibration contains a checkerboard pattern with 9×6 squares of 3.5 cm edge length each.

The pattern also contains two black corners opposite two white corners. This criteria serves to determine the orientation of the pattern. This also enables automatic identific-ation of the correspondence between reference points on the model plane and square corners in images. As imperfections on the pattern surface can affect the accuracy of the calibration, the checkerboard is attached to a flat surface. The used

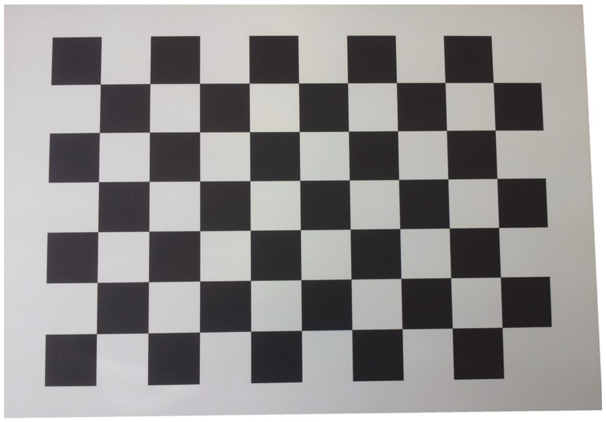


Figure 8.44: The checkerboard pattern plane used for camera calibration. It contains a pattern with 9×6 squares of 3.5 cm edge length each and two black corners opposite two white corners.

In total, 23 images of the plane were taken. Twelve of them (called Set A) were taken at close range, while the other 11 (called Set B) were taken at a larger distances. We applied the calibration algorithm to the whole set (A and B). we used the utility QCamCalib[2] from the the Robot Construction Kit [ROCK, 2015] to calibrate the camera and MATLAB to visualize the results. The calibration results are shown in Figure 8.45 and Figure 8.46.

We can evaluate calibration accuracy by examining the reprojection errors and the camera extrinsics. To evaluate the calibration results, the reprojection errors are calculated.

[2]`http://blog.rock-robotics.org/post/75801526956/qcamcalib-pinhole-camera-calibration`

They are the distances in pixels between the detected and the reprojected points. We calculated them by projecting the checkerboard points from world coordinates (defined by the checkerboard), into image coordinates. These reprojected points are compared to the corresponding detected points. As a rule of thumb, reprojection errors should be less than one pixel. The reprojection errors displayed in Figure 8.45 as a bar graph and as a scatter plot. The scatter plot displays the reprojection errors for each point. A good calibration typically results in a compact cloud of points. Outliers indicate potential issues with the corresponding images. The 3D extrinsic parameters plot (shown in Figure 8.46) provides a camera-centric view of the patterns.

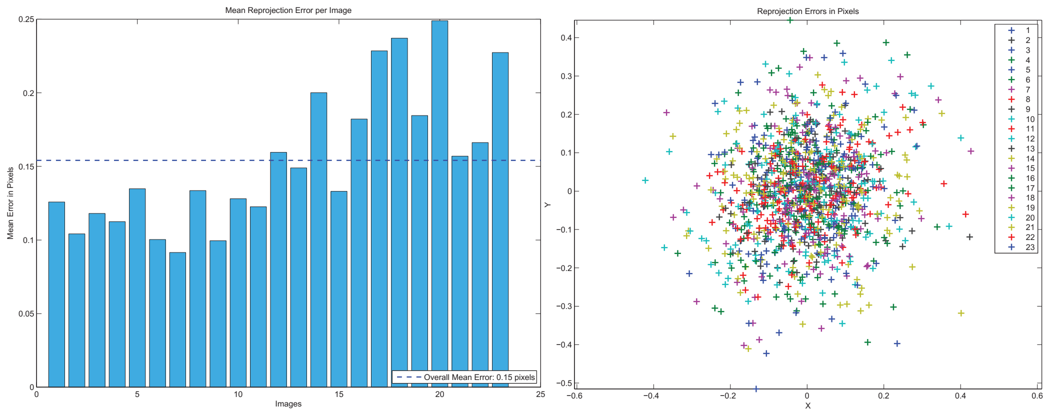


Figure 8.45: Camera calibration reprojection errors displayed as a bar graph (left) and as a scatter plot (right). Both of these visuals are expressed in pixels. The bar graph displays the mean reprojection error per image along with the overall mean error. The bar labels correspond to the image IDs. The highlighted bar corresponds to the selected image.

Intrinsic camera calibration parameter matrix (**K**) and Radial Distortion ($\mathbf{R_d}$) are:

$$\mathbf{K} = 1.0e+03 \begin{bmatrix} 2.8467 & 0 & 0 \\ 0 & 2.8486 & 0 \\ 1.6160 & 1.2857 & 0.001 \end{bmatrix} \tag{8.4}$$

$$\mathbf{R_d} = [0.0747 \quad -0.4442 \quad 0.5998] \tag{8.5}$$

From the reprojection error results, we have an overall mean error of 0.15 pixels with all error results scattered roughly within a circle of 0.4 pixels in radius. Intuitively, The camera parameters were estimated consistently for the two sets of images and therefore can be reliably used for the further calculations.

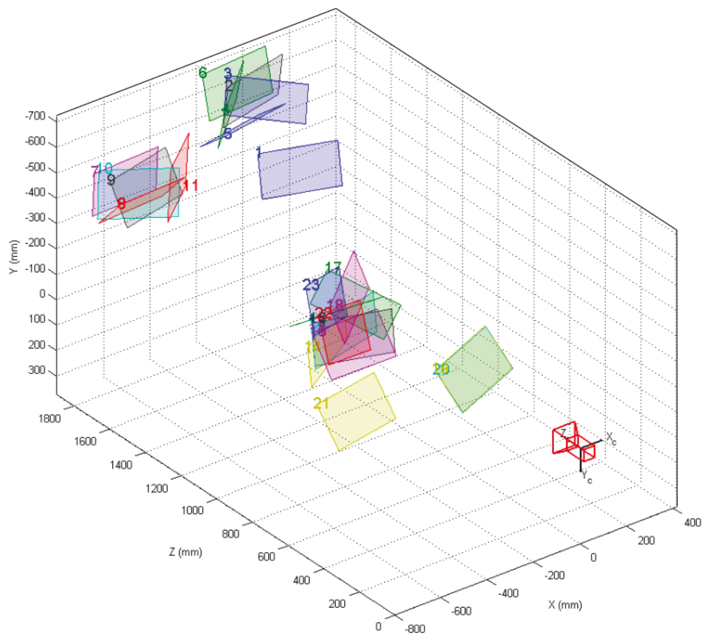


Figure 8.46: Extrinsic parameters visualization (camera centric). The numbers attached to the frames are the image index for the images used for the calibration.

8.7 Path Planning Experiments

The experiments presented here aim to verify the validity and impact of the whole approach, in the context of a natural environment. Here, natural means that the environment characteristics are not known a priori.

In this regard, the robot behavioral influence based on the knowledge available and learned from previous experiments will be utilized during the tests. Therefore, the path is planned according to the prediction of the terrain ahead of the robot. The main settings of these tests are depicted in Figure 8.47.

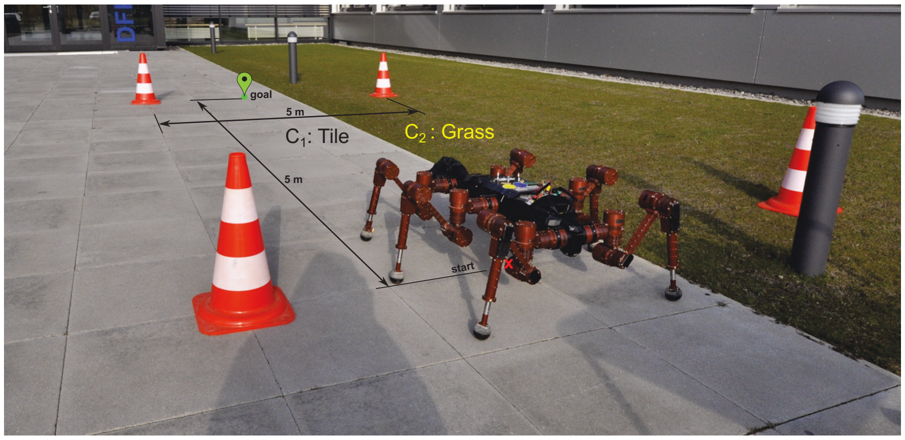


Figure 8.47: Experiment setup to test with the *i*ARTEC proposed architecture.

In the first set of experiments, the robot is commanded to reach a waypoint five meters ahead of its starting position. As depicted in Figure 8.48, this is to move from the start position (S) to the goal position (G).

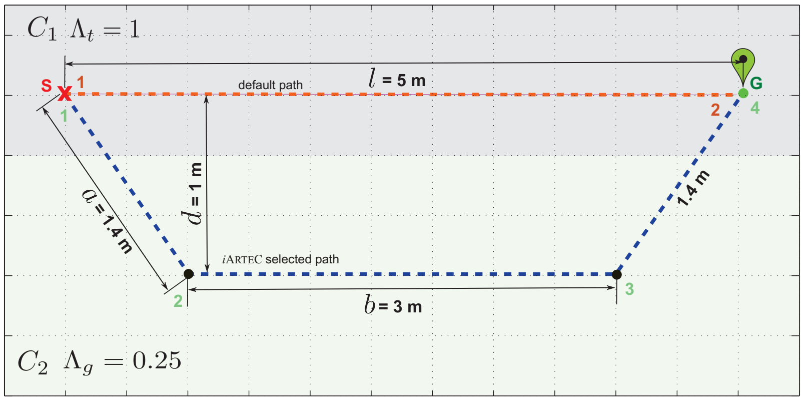


Figure 8.48: Sketch of the experiments for testing with the proposed architecture.

Path Planning Experiment without *i*ArteC

With *i*ARTEC disabled, the robot planner generates a straight, direct trajectory as the shortest path to its goal. The results of this normal behavior are shown in Figure 8.49.

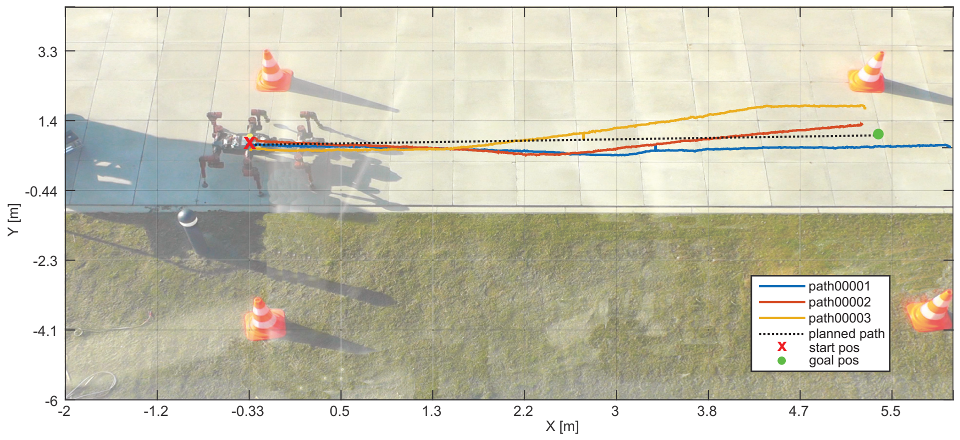


Figure 8.49: Results of three runs of the robot without applying *i*ARTEC to the control of its path planning. The actual trajectories followed by the robot are very similar and close to the default one.

As can be noticed from the generated trajectory, the robot aims for its goal and successfully reached it. In addition, the straight trajectory would drive the robot on tile only. The test is also repeated three times to have more reliable results and the robot drives at a speed of 0.03 m/s.

Path Planning Experiment with *i*ArteC

To test the effect of the proposed architecture on the robot behavior, the same experiment is repeated but with *i*ARTEC enabled. In *i*ARTEC , the adaptive behavior is driven by MTCN whose characteristics are assumed to be known and provided by the user. This implies that we need to choose the values of Λ for each terrain class. This choice also depends on the test environment.

For the following experiments, aiming towards showing the impact of the overall method on the robot control, we chose to minimize the amount of mechanical vibrations within the robot structure. Thus, two metrics are defined here, one corresponding to the tile class and the other for the grass as:

$$\begin{aligned}\Lambda_t &= 1,\\ \Lambda_g &= 0.25\end{aligned} \tag{8.6}$$

where Λ_t and Λ_g are the MTCN for tile and grass, respectively. In addition, the two terrains as assumed in Section 7.5.1 are traversable (i.e. $\tau = 1$). Those two metrics are used in the next tests to show the impact of the proposed architecture on the robot behavior.

From the sketch in Figure 8.48, the robot at the start position detects the grass terrain by its long–range visual classifier as illustrated in Figure 8.51. The two path ratios are calculated according to Equation 7.7 and Equation 7.8 as:

$$\alpha = \frac{\Lambda_t}{\Lambda_g} = 4,$$

$$\beta = \frac{l}{d} = 5$$

As $\alpha > 1$ and $\beta > 2$ then the condition in Equation 7.21 is satisfied.Therefor, *i*ARTEC decides that the limit between terrain classes will actively influence the robot trajectory. This is obvious in the results given in Figure 8.50 as the robot path planner favors grass over tile terrain. It generates an alternative path that goes over grass terrain and follows it till it reaches its goal. The resulting trajectory has a slightly longer length but the vibrations within the chassis are significantly reduced when possible.

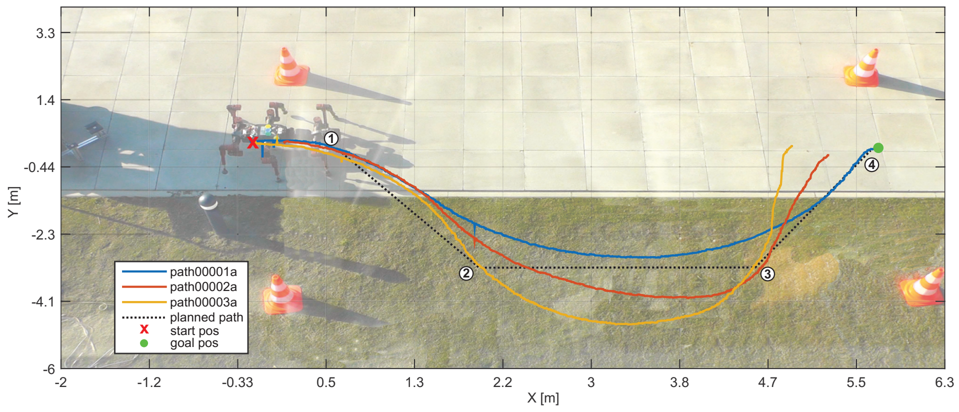


Figure 8.50: Experiment results of three runs to test with the proposed architecture (*i*ARTEC) applied to the robot. The actual trajectories followed by the robot clearly show that the robot behavior is influenced by the predicted MTCN metrics.

The differences in the robot actual trajectories in each experiment (plotted in Figure 8.49 and Figure 8.50) are due to the errors in odometry model which were delivered by the robotic framework ROCK and therefore could not be avoided.

The results of the long–range visual terrain classifier during the experiment at different key positions are given in Figure 8.51. The presented images are taken at the key points marked in Figure 8.50.

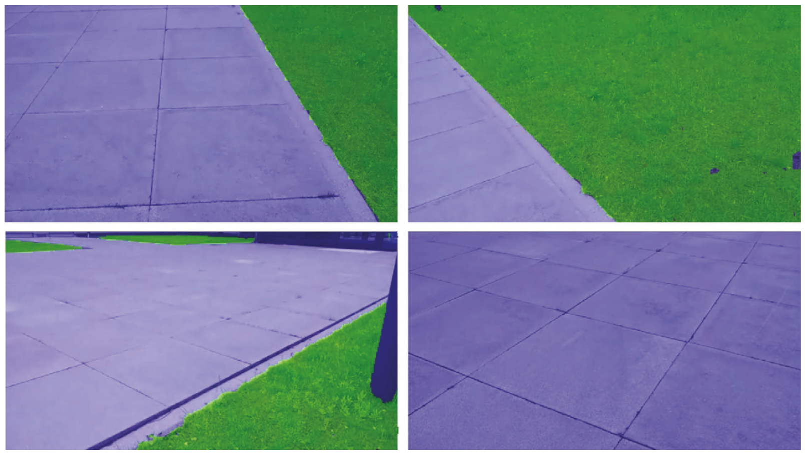


Figure 8.51: Long range terrain classification results during the experiment at different key positions. The presented images are taken at the key points marked in Figure 8.50 at start point 1 (upper left), at first turning point 2 (upper right), at second turning point 3 (down left), and at goal point 4 (down right).

8.8 Summary

In the first part of this chapter, a detailed description of the experimental setup and results from the experiments preformed to study the behavior and characteristics of the foot–ground contact forces are presented. Also an account of the results of the experiments conducted to test and verify the Plastic Terramechanics Particle (PTP) model is given.

The experiment and simulation results show that through the right choice of parameters, the presented foot–ground contact model provides a very close results to the vertical forces obtained in the real experiments especially at the start and end of the contact as evident in the test and verification experiments. Moreover, the model results even excellently reproduces the force response characteristics.

A comparison between the lateral forces in real experiments and the model in simulation-shows that the PTP approach is able to make realistic predictions in principle, although not as good as that could be reached in the case of the normal force (see Figure 8.26). The force produced by the model is higher than that of the reference experiments with constant lateral force (see Figure 8.25). This is mainly due to the unmodeled bulldozing effect that occurs when a special combination of foot, soil and velocity is met. This effect results in the accumulation of soil particles in front of the foot that generates a resistance force (mainly in the lateral direction).

In the second part of this chapter, experiments with the long–range visual terrain classifier are presented. The first set of experiments were to select the classifier parameters that gave the most desirable mixture of accuracy and performance. In the second set of experiments, we tested the optimized classifier. The results showed that our methods effectively classified the terrains in both homogeneous and heterogeneous terrain images with high accuracy (with an average of 96% in both cases).

The final part of this chapter,the experiments with the robot showed good results and the robot behaves as expected when the *i*ARTEC architecture is applied for the overall control of the robot movement. The experiments presented here combines both the prediction aspects of the already learned terrain types. The predicted terrains influence the planned path of the robot to reach its goal position.

The true function of philosophy is to educate us in the principles of reasoning and not to put an end to further reasoning by the introduction of fixed conclusions.
— George Henry Lewes, (English philosopher and critic of literature, 1817–1878)

9 Conclusions and Future Outlook

Legged mobile robots are confronted by real, unengineered environment which makes it difficult or impossible to foresee the interaction between the robot and the terrain. This is especially true in the context of exploration robots, which aim to investigate unknown regions. These robots navigate unreliably over uneven terrain –even when designed with inherent stability– mostly because they do not take into account the type of terrain they walk on and a lack of good models for ground interaction.

While several other works concentrate on being able to predict the traversability of the environment, the present research focuses on a complementary problem, modeling and predicting the interaction forces between the robot feet and the ground in direct contact with them and classifying the terrain type in the near and far field of the robot.

The work presented in this dissertation resulted in an architecture called "Intelligent Architecture for Legged Robot Terrain Classification Using Proprioceptive and Exteroceptive Data (*i*ARTEC)". The proposed architecture is composed mainly of four subsystems: foot-soil interaction, local terrain classification, remote terrain visual classification, and path planning and following.

Firstly, it allows the robot to estimate the interaction forces between its feet and the terrain and to classify local and remote terrain, as well as the relations linking those representations. This learning mechanism is performed according to the robot own sensors data.

Secondly, based on the knowledge available, *i*ARTEC makes use of the robot terrain interaction with the local terrain to predict the most probable values for the remote ones. Based on a predefined metric, MTCN, the performance of the robot on a given terrain is predicted. This allows the navigation of the rover to be influenced by the models learned online.

Implementing these subsystems into standalone modules with a defined interfaces makes the architecture very flexible for addition, modification, and/or replacement of required new functionality. *i*ARTEC was successfully implemented and tested on the CREX robot platform. Many experiments and tests were conducted to show the various capabilities and limitations of the approach.

We also argue that it seems more appropriate for a robot to classify terrains based on its needs and according to how they affect its behavior rather than based on human defined classes of only traversable and nontraversable terrain. In other words, our approach categorizes the terrain in a way that is suitable for the robot path planning as presented in our terrain dependent path planning experiment results.

Finally, another essential point is that although our classification is fixed in a rigid number of classes, it is rather modular and expandable with minor modifications. The terrain classes ares actually learned and can evolve with respect to the situation expected to encounter the robot. Therefore, the approach is very flexible and is capable of incorporating new types of terrain or interaction models. Also note that the present work does not take the traversability (as in the navigation-related notation) into account since the focus lies on the terrain type learning and prediction.

9.1 Main Achievements

In our approach, complex soil dynamics are modeled using volume–cells containing PTPs. It considerably reduces the modeling complexity as well as to provide real-time capability. Moreover, in order to calculate realistic soil reaction forces, pressure look up tables of sinkage and lateral movement are used that can be easily measured through simple soil experiments. Thus, complex soil parameters are unnecessary in our approach.

Recent work in terrain classification has relied largely on 3D sensing methods and color based classification. We present an approach that works with a single, compact camera and maintains high classification rates that are robust to changes in illumination. Terrain is classified using a bag of visual words (BoW) created from speeded up robust features (SURF) with a support vector machine classifier. We present several novel techniques to augment this approach. A gradient descent inspired algorithm is used to adjust the SURF Hessian threshold to reach a nominal feature density. A sliding window technique is also used to classify mixed terrain images with high resolution. We demonstrate that our approach is suitable for legged robots by performing offline and real-time terrain classification on the CREX robot. The classifier result of the predicted terrain type is used to decide whether to select another path for traversing terrain of varying difficulty.

Also, we thoroughly investigated the applicability of local descriptors for visual terrain classification on legged mobile robots. Many of the current texture classification approaches use sharp images containing mostly a single texture captured from a fixed camera angle under controlled conditions. We used images from real runs of the robot containing blurred images and consisting of multiple terrains. We used the state of the art legged robot CREX for this purpose. The CREX robot can move over rough terrain at relatively fast speeds. So the images captured by this robot contain blur and varying light conditions and changes in terrain types.

We used the SURF image descriptors calculated at keypoints on a grid drawn across the image. This is against the traditional way of using them, where they are mostly required to find their own keypoints. The modified version calculates other descriptors at specific grid cell sizes with much good performance. In addition SURF has one of the smallest feature vectors and is fast to train.

Hence, we have demonstrated that visual terrain classification can be performed at different resolutions using SURF image descriptors on color images. Furthermore, it is demonstrated that visual terrain classification can be successfully performed on an outdoor driving robot even in non optimal conditions, such as motion blur induced by a fast moving robot and its vibrating camera, different weather conditions, both wet and dry ground surfaces and a low camera viewpoint. We used images from real runs of the robot containing blurred images with non-sharp terrain boundaries.

Feature extraction was improved with our gradient descent inspired detection algorithm. Extracting a target number of features increased classification accuracy while decreasing memory requirements. A sliding window classifier was also designed to identify patches in heterogeneous terrain images. This technique showed promising offline results, but struggled slightly with terrain boundaries.

In addition, we also demonstrated the effectiveness of our feature-based terrain classification in offline and real-time testing. Offline experiments provided valuable data on classification performance in different experiment setups in a controlled environment. The findings allowed us to select parameters that gave the optimal mixture of accuracy and performance. Real-time testing showed that our methods effectively assisted autonomous navigation on the CREX platform. The robot was able to traverse terrain with better *characteristics* with classification-in-the-loop despite the longer distance it has to travel.

The different possible paths can be more or less efficient according to the robot interaction dynamics with the terrains it traverses. The research presented here proposes an approach allowing the robot–terrain interaction characteristics, not only the traversability, to be asserted based on remote data. Based on this knowledge, the path with the best MTCN can be used.

The proposed “Metric of Terrain Class Navigation” (MTCN) is a global metric characterizing the robot terrain interaction and take into account the distance traveled. Thus it would be possible to learn if the additional distance that must be traveled to reach a better terrain is worth traversing.

For the low-level control of the robot actuators to follow the desired trajectory, we designed an effective proportional input-scaling controller. This controller utilizes a chained form transformation to accomplish its closed-loop control strategy.

Through realistic simulations and field experiments, it is shown that the designed path planner and follow controller are robust and converges giving good performance provided that the robot starts sufficiently close to the desired path.

The designed path-follow controller model has the advantages that it can be directly used for differential robots with a trailer and it can be generalized to robot with N-trailers in a straightforward manner. We extended the path-following controller proposed in this thesis to be used with hybrid wheel–legged and car-like robots. Examples of these extended controller are presented in [Ahmed et al., 2014c] for the hybrid wheel–legged rover Coyote II[1] and in [Birnschein et al., 2014] and [Ahmed and Yüksel, 2013] for the new concept EOscc2[2] car.

Overall, we demonstrated that our terrain classification architecture can be effectively used in real-time to assist legged robot navigation.

Architecture Requirements

From a user point of view, deploying the proposed architecture is relatively easy but a few elements have to be provided as inputs. The list below summarizes the requirements of the approach:

- The proposed architecture is modular, so it can be implemented as a whole or just the desired components can be utilized.
- The MTCT metric must be given for each terrain class. The navigation of the robot is optimized based on the MTCT metric hence the rule driving this optimization has also to be given. This means defining the notion of *good* and *bad* with respect to the metric, or in other words, whether a small or a big MTCT value is preferred.
- The last requirement is to define at what extent the trajectory followed by the robot can deviate from its default trajectory. A default trajectory consists of the robot evolving without taking into consideration the MTCT value. This requirement is fulfilled by defining the parameters α and β (see Section 7.4).

9.2 Future Work

The experiments of the CREX robot showed that the limitation of the camera field of view, as well as the limitation resulting from the camera horizon lead to strong limitation of the impact of remote terrain classification. In this prospect, it would be interesting to use the approach combining other sensory sources, such as multi-agent cooperation. For example, a flying robot could provide a bird-eye view of the terrains, or multiple robots working together could improve the perception quality of the terrain.

Additionally, the robot, through our architecture, used the gained knowledge of the terrain type to autonomously modify (or even select a different) path in favor of traversing over a *better* terrain. A complementary work to this would be to further benefit from this

[1]`http://robotik.dfki-bremen.de/en/research/robot-systems/coyote-ii-1.html`
[2]`http://robotik.dfki-bremen.de/en/research/robot-systems/eo-smart-connecting-3.hml`

knowledge in optimizing other robotic behaviors in context with specific terrain types, e.g., selecting a particular gait for each terrain type.

Another essential point is that in our architecture the terrain classes are fixed as they must be learned beforehand. To push forward the robot capabilities to learn on its own it would be interesting to use online learning. Thus, a learning mechanism would be employed when enough samples of the class “unknown” are available. We belive that self-supervised learning would be a direct extension to our work.

To improve the reliability of the foreseen learning mechanism, it could rely on the assumption that the learning occurs faster than changes in the environment, and therefore, that the samples belong to the same terrain class. This assumption can be removed by adding a clustering step before the learning mechanism. Thus, having this additional step would allow to classify the “unknown” samples in different classes.

Finally, we believe that the methods proposed in this thesis can likely also be implemented on other robot types such as wheeled robots, although we did not test this option in our work. It would be very interesting to try them on a robotic platform of the other types

List of Symbols and Abbreviations

*i*ARTEC	Intelligent Architecture for Legged Robot Terrain Classification Using Proprioceptive and Exteroceptive Data, [4]
BoW	bag of words, [64]
CA	Cellular Automata, [20]
CREX	Crater Explorer, a six-legged walking robot that is based on the developments in the project SpaceClimber., [11]
DEM	Discrete (Distinct) Element Models, [20]
FEM	Finite Element Models, [20]
LIDAR	Light Detection And Ranging, [21]
MTCN	Metric of Terrain Class Navigation, [69]
PTP	Plastic Terramechanics Particle model, [42]
SpaceClimber	A Semi-Autonomous Free-Climbing Robot for the Exploration of Crater Walls and Bottoms, [31]
SURF	Speeded Up Robust Features, [57]

Bibliography

[Ahmed, 2014] Ahmed, M. (2014). Car of the future: Innovative technologies in electromobility for development, design, and construction of smart cars. Talk presented at HANNOVER MESSE 2014, Forum Robotics, Automation & Vision. [8]

[Ahmed and Babu, 2014] Ahmed, M. and Babu, A. (2014). Autonomous steering controller for path following. In *Proceedings of the RIC Project Day Workgroups "Framework & Standardization" and "Manipulation & Control". RIC Project Day, June 19, Bremen, Germany*, volume 14-05 of *DFKI Documents, D*, pages 118–119. DFKI Robotics Innovation Center Bremen, Selbstverlag. [8]

[Ahmed et al., 2010a] Ahmed, M., Benitez, L. V., and Kirchner, F. (2010a). Accurate identification and simulation of brushless DC drive actuating system for high performance applications. In *The 4th Int'l Industrial Control & Automation Technology Exhibition and Conference. Automation Technology Egypt 2010 (Automation -2010), May 10-12, Cairo, Egypt*. Automation 2010. [11]

[Ahmed et al., 2015] Ahmed, M., Ebrahim, M. A., Ramadand, H. S., and Becherif, M. (2015). Optimal genetic-sliding mode control of vsc-HVDC transmission systems. *Energy Procedia*, 69. International Conference on Technologies and Materials for Renewable Energy, Environment and Sustainability, TMREES15. [11]

[Ahmed et al., 2014a] Ahmed, M., Eich, M., and Bernhard, F. (2014a). Design and control of mira: a lightweight climbing robot for ship inspection. In *World Symposium on Mechatronics Engineering & Applied Physics (WSMEAP2014). International Conference on Mechatronics Engineering (ICME-2014)*, pages 58–62, Sousse, Tunisia. IEEE. [11]

[Ahmed and Kirchner, 2009] Ahmed, M. and Kirchner, F. (2009). A simulation environment to be utilised in the design and test process of the HEVs and EVs BLDC drive and its control. In *11th European Regional Conference of the International Society of Terrain-Vehicle Systems (ISTVS-09)*, Bremen, Germany. ISTVS'09. [11]

[Ahmed and Kirchner, 2014] Ahmed, M. and Kirchner, F. (2014). Design and implementation of a long range visual terrain classifier for legged robots. In Sammouda, R., editor, *World Symposium on Computer Applications & Research WSCAR' 2014, International Conference on Signal Processing and Remote Sensing (ICSPRS2014)*, pages 117–122, Sousse, Tunisia. IEEE. [11]

[Ahmed et al., 2014b] Ahmed, M., Oekermann, C., and Kirchner, F. (2014b). Cosimulation environment for mechanical design optimization with evolutionary algorithms. In Sammouda, R., editor, *World Symposium on Computer Applications & Research WSCAR' 2014, International Conference on Artificial Intelligence (ICAI' 2014)*, pages 21–26, Sousse, Tunisia. IEEE. [11]

[Ahmed et al., 2011] Ahmed, M., Quack, L., Langosz, M., and Yoo, Y.-H. (2011). Development of a real and simulation testbed for legged robot soil interaction. In *International Conference of the International Society for Terrain-Vehicle Systems, (ISTVS-11)*, pages 110–116, Blacksburg, Virginia, USA. International Society for Terrain-Vehicle Systems, ISTVS2011. [11, 92]

[Ahmed et al., 2014c] Ahmed, M., Sonsalla, R., and Kirchner, F. (2014c). Autonomous path tracking steering controller for extraterrestrial terrain exploration rover. In *40th COSPAR Scientific Assembly*, Moscow, Russian Federation. cosmos. [11, 136]

[Ahmed and Yoo, 2010] Ahmed, M. and Yoo, Y.-H. (2010). Measurement and control of the contact forces between walking robot legs and its environment. In *Proceedings of the Joint 9th Asia-Pacific ISTVS Conference and Annual Meeting of Japanese Society for Terramechanics*, Sapporo, Japan. International Society for Terrain-Vehicle Systems, ISTVS2010. [11]

[Ahmed et al., 2010b] Ahmed, M., Yoo, Y.-H., and Kirchner, F. (2010b). A co-simulation framework for design, test and parameter optimization of robotic systems. In *ISR / ROBOTIK 2010. The joint conference of the 41st International Symposium on Robotics and the 6th German Conference on Robotics (ISR/ROBOTIK-2010), June 7-9, Munich, Germany*. ISR/ROBOTIK2010. [11, 123]

[Ahmed and Yüksel, 2013] Ahmed, M. and Yüksel, M. (2013). Autonomous path tracking steering controller for EO smart connecting car. In Tagoug, N., editor, *Proceeding of the World Congress on Multimedia and Computer Science 2013. International Conference on Intelligent Automation and Robotics (ICIAR-13)*, pages 45–50, Hammamet, Tunisia. IEEE. [11, 136]

[Arthur and Vassilvitskii, 2007] Arthur, D. and Vassilvitskii, S. (2007). K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pages 1027–1035, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics. [62]

[ATI URL, 2014] ATI URL (2014). Industrial Automation (ATI): Net F/T sensor. `http://www.ati-ia.com/products/ft/ft_NetFT.aspx`, *laste visited 02.03.2014*. [88]

[Bay et al., 2008] Bay, H., Ess, A., Tuytelaars, T., and Gool, L. V. (2008). Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346–359. [61]

[Bekker, 1960] Bekker, M. (1960). Introduction to terrain-vehicle systems. *University of Michigan Press, Ann Arbor*. [14, 20, 37, 39, 41]

[Birnschein et al., 2014] Birnschein, T., Kirchner, F., Ahmed, M., Yueksel, M., Yoo, Y.-H., Oekermann, C., Girault, B., Kroffke, S., and Gruenwald, D. (2014). Enhancing mobility using innovative technologies and highly flexible autonomous vehicles. In *18th International Forum on Advanced Microsystems for Automotive Applications (AMAA 2014): Smart Systems for Safe, Clean, and Automated Vehicles*, Berlin, Germany. accepted. [11, 136]

[Brooks and Iagnemma, 2005] Brooks, C. and Iagnemma, K. (2005). Vibration-based terrain classification for planetary exploration rovers. *IEEE Transactions on Robotics*, 21(6):1185–1191. [15, 16, 18, 24]

[Brooks and Iagnemma, 2007] Brooks, C. and Iagnemma, K. (2007). Self-supervised classification for planetary rover terrain sensing. In *Aerospace Conference, 2007 IEEE*, pages 1–9. [16, 17, 24]

[Bubeck et al., 2012] Bubeck, S., Meila, M., and von Luxburg, U. (2012). How the initialization affects the stability of the k-means algorithm. *ESAIM: Probability and Statistics*, PS 16:436–452. [62]

[Burgard and Hebert, 2008] Burgard, W. and Hebert, M. (2008). World modeling. In Siciliano, B. and Khatib, O., editors, *Handbook of Robotics*, chapter 36, pages 853–869. Springer. [69]

[Burges, 1998] Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Discov.*, 2(2):121–167. [52]

[Byun and Lee, 2002] Byun, H. and Lee, S.-W. (2002). Applications of support vector machines for pattern recognition: A survey. In Lee, S.-W. and Verri, A., editors, *Pattern Recognition with Support Vector Machines*, volume 2388 of *Lecture Notes in Computer Science*, pages 213–236. Springer Berlin Heidelberg. [49]

[Chung et al., 2008] Chung, W., Fu, L., and Hsu, S. (2008). Motion control. In Siciliano, B. and Khatib, O., editors, *Handbook of Robotics*, chapter 7, pages 133–159. Springer. [77]

[Collins and Coyle, 2008] Collins, E.G., J. and Coyle, E. (2008). Vibration-based terrain classification using surface profile input frequency responses. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 3276–3283. [18]

[Cortes and Vapnik, 1995] Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Mach. Learn.*, 20(3):273–297. [49]

[Coyle et al., 2008] Coyle, E., Collins Jr, E. G., DuPont, E., Ding, D., Wang, H., Cooper, R. A., and Grindle, G. (2008). Vibration-based terrain classification for electric powered wheelchairs. In *Telehealth/AT 2008 Proceedings of the IASTED International Conference on Telehealth/Assistive Technologies*, pages 139–144. [17, 18]

[Craig, 2004] Craig, J. J. (2004). *Introduction to Robotics: Mechanics and Control*. Prentice Hall, Boston, MA, USA, 3rd edition. [85]

[de Gea Fernández et al., 2009] de Gea Fernández, J., Lemburg, J., Roehr, T. M., Wirkus, M., Gurov, I., and Kirchner, F. (2009). Design and control of an intelligent dual-arm manipulator for fault-recovery in a production scenario. In *IEEE Conference on Emerging Technologies and Factory Automation. IEEE Conference on Emerging Technologies and Factory Automation (ETFA-09), September 22-25, Mallorca, Spain*. [88]

[DFKI Robot Test Track, 2015] DFKI Robot Test Track (2015). Outdoor testing facility for robot benchmarking. `http://robotik.dfki-bremen.de/en/research/research-facilities/robot-test-track.html`, *laste visited 08.02.2015*. [107]

[Duan and Keerthi, 2005] Duan, K.-B. and Keerthi, S. (2005). Which is the best multiclass svm method? an empirical study. In Oza, N., Polikar, R., Kittler, J., and Roli, F., editors, *Multiple Classifier Systems*, volume 3541 of *Lecture Notes in Computer Science*, pages 278–285. Springer Berlin Heidelberg. [53]

[Dupont et al., 2008] Dupont, E. M., Moore, C. A., Collins, Jr., E. G., and Coyle, E. (2008). Frequency response method for terrain classification in autonomous ground vehicles. *Autonomous Robots*, 24(4):337–347. [15, 16, 18]

[DuPont et al., 2006] DuPont, E. M., Roberts, R. G., and Moore, C. A. J. (2006). Speed independent terrain classification. In *Proceeding of the Thirty-Eighth Southeastern Symposium on System Theory (SSST '06)*, pages 240–244. [16, 18]

[Fawzi et al., 2014] Fawzi, A., Davies, M. E., and Frossard, P. (2014). Dictionary learning for fast classification based on soft-thresholding. *CoRR*, abs/1402.1973. [53, 64]

[Garcia Bermudez et al., 2012] Garcia Bermudez, F., Julian, R. C., Haldane, D. W., Abbeel, P., and Fearing, R. S. (2012). Performance analysis and terrain classification for a legged robot over rough terrain. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2012)*, pages 513–519. IEEE. []

[Giguère and Dudek, 2009] Giguère, P. and Dudek, G. (2009). Clustering sensor data for autonomous terrain identification using time-dependency. *Auton. Robots*, 26(2-3):171–186. [15]

[Hall, 2014] Hall, D. S. E. (2014). Dfki space exploration hall, crater area. `http://robotik.dfki-bremen.de/en/research/research-facilities/space-exploration-hall.html`, *laste visited 02.03.2014*. [99]

[He et al., 2014] He, J., Marin, A., and Ostendorf, M. (2014). Effective data-driven feature learning for detecting name errors in automatic speech recognition. In *Spoken Language Technology Workshop (SLT), 2014 IEEE*, pages 230–235. [60]

[Hsu and Lin, 2002] Hsu, C.-W. and Lin, C.-J. (2002). A comparison of methods for multiclass support vector machines. *Neural Networks, IEEE Transactions on*, 13(2):415–425. [53]

[Irani et al., 2013] Irani, R., Bauer, R., and Warkentin, A. (2013). Dynamic wheel-soil model for lightweight mobile robots with smooth wheels. *Journal of Intelligent & Robotic Systems*, 71(2):179–193. []

[Ishigami et al., 2007] Ishigami, G., Miwa, A., Nagatani, K., and Yoshida, K. (2007). Terramechanics-based model for steering maneuver of planetary exploration rovers on loose soil. *Journal of Field Robotics*, 24(3):233–250. [38]

[Janosi and Hanamoto, 1961] Janosi, Z. and Hanamoto (1961). Analytical determination of drawbar pull as a function of slip for tracked vehicles in deformable soils. In *In Proc. the First International Conference on Terrain-Vehicle Systems, Edizioni Minerva Tecnica, Torino.* [41]

[Khan et al., 2012] Khan, Y., Masselli, A., and Zell, A. (2012). Visual terrain classification by flying robots. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 498–503. [66]

[Kim et al., 2010] Kim, K., Ko, K., Kim, W., Yu, S., and Han, C. (2010). Performance comparison between neural network and svm for terrain classification of legged robot. In *SICE Annual Conference 2010, Proceedings of*, pages 1343–1348. []

[Krebs et al., 2009] Krebs, A., Pradalier, C., and Siegwart, R. (2009). Comparison of boosting based terrain classification using proprioceptive and exteroceptive data. In Khatib, O., Kumar, V., and Pappas, G., editors, *Experimental Robotics*, volume 54 of *Springer Tracts in Advanced Robotics*, pages 93–102. Springer Berlin Heidelberg. []

[Krell et al., 2013] Krell, M. M., Straube, S., Seeland, A., Wöhrle, H., Teiwes, J., Metzen, J. H., Kirchner, E. A., and Kirchner, F. (2013). pySPACE — a signal processing and classification environment in Python. *Frontiers in Neuroinformatics*, 7(40). [54, 55, 56]

[Krenn and Hirzinger, 2009] Krenn, R. and Hirzinger, G. (2009). SCM: A soil contact model for multi-body system simulations. In *Proceedings of the 11th European Regional Conference of the ISTVS 2009I*, Bremen, Germany. [21]

[Langosz et al., 2011] Langosz, M., Ahmed, M., Quack, L., and Yoo, Y.-H. (2011). Modeling of leg soil interaction using genetic algorithms. In *International Conference of the International Society for Terrain-Vehicle Systems, (ISTVS-11)*, pages 110–116. International Society for Terrain-Vehicle Systems, ISTVS2011. [11]

[Liu and Wong, 1996] Liu, C. and Wong, J. (1996). Numerical simulations of tire-soil interaction based on critical state soil mechanics. *Journal of Terramechanics*, 33(5):209–221. [21]

[Lowe, 2004] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110. [61]

[Mangasarian, 1999] Mangasarian, O. (1999). Arbitrary-norm separating plane. *Operations Research Letters*, 24(1-2):15–23. [51]

[Meirion-Griffith and Spenko, 2014] Meirion-Griffith, G. and Spenko, M. (2014). Simulation and experimental validation of a modified terramechanics model for small-wheeled vehicles. *International Journal of Vehicle Design*, Volume 64(2):153–169. [45]

[Murray and Sastry, 1991] Murray, R. and Sastry, S. (1991). Steering nonholonomic systems in chained form. In *Decision and Control, 1991., Proceedings of the 30th IEEE Conference on*, pages 1121–1126 vol.2. [80]

[Nakashima and Oida, 2004] Nakashima, H. and Oida, A. (2004). Algorithm and implementation of soil–tire contact analysis code based on dynamic fe–de method. *Journal of Terramechanics*, 41(2–3):127 – 137. 14th International Conference of the {ISTVS}. [21]

[Nakashima et al., 1990] Nakashima, H., Tanaka, T., and Yamazaki, M. (1990). Finite element analysis of soil-lug interaction. In *Proceedings of the 10th int conf of the ISTVS*, volume 1, pages 277–8, Kobe, Japan. [21]

[Nayar et al., 2008] Nayar, H., Balaram, B., Cameron, J., Jain, A., Lim, C., Mukherjee, R., Peters, S., Pomerantz, M., Reder, L., Shakkottai, P., and Wall, S. (2008). A lunar surface operations simulator. In Carpin, S., Noda, I., Pagello, E., Reggiani, M., and von Stryk, O., editors, *Simulation, Modeling, and Programming for Autonomous Robots*, volume 5325 of *Lecture Notes in Computer Science*, pages 65–74. Springer Berlin Heidelberg. [20]

[Nayar et al., 2009] Nayar, H., Jain, A., Balaram, J., Cameron, J., Lim, C., Mukherjee, R., Pomerantz, M., Reder, L., Myint, S., Serrano, N., and Wall, S. (2009). Recent developments on a simulator for lunar surface operations. In *AIAA SPACE 2009 Conference and Exposition, Pasadena, CA, September 14-17*. [20]

[Ojeda et al., 2006] Ojeda, L., Borenstein, J., Witus, G., and Karlsen, R. (2006). Terrain characterization and classification with a mobile robot. *Journal of Field Robotics*, 23:103–122. [16, 18]

[Oriolo et al., 2002] Oriolo, G., De Luca, A., and Vendittelli, M. (2002). WMR control via dynamic feedback linearization: design, implementation, and experimental validation. *Control Systems Technology, IEEE Transactions on*, 10(6):835–852. [77]

[Oyallon and Rabin, 2013] Oyallon, E. and Rabin, J. (2013). An analysis and implementation of the SURF method, and its comparison to SIFT. *Image Processing On Line*, -:1–31. [61]

[Philippsen, 2007] Philippsen, R. (2007). E* interpolated graph replanner. In *Workshop Proceedings on Algorithmic Motion Planning for Autonomous Robots in Challenging Environments, held in conjunction with the IEEE International Conference on Intelligent Robots and Systems (IROS)*. [69]

[Philippsen et al., 2007] Philippsen, R., Jensen, B., and Siegwart, R. (2007). Towards real-time sensor-based path planning in highly dynamic environments. In Laugier, C. and Chatila, R., editors, *Autonomous Navigation in Dynamic Environments*, volume 35 of *Springer Tracts in Advanced Robotics*, pages 135–148. Springer Berlin Heidelberg. [69]

[Pla-Castells et al., 2006] Pla-Castells, M., García-Fernández, I., and Martínez, R. J. (2006). Interactive terrain simulation and force distribution models in sand piles. In *Cellular Automata*, pages 392–401. Springer. [22]

[pySPACE, 2015] pySPACE (2015). pyspace is a signal processing and classification environment written in python pySPACE web site. `http://pyspace.github.io/pyspace` , *laste visited 02.03.2015*. [54]

[Reece, 1965] Reece, A. R. (1965). Principles of soil-vehicle mechanics. *Proceedings of the Institution of Mechanical Engineers: Automobile Division*, 180(1):45–66. [39]

[ROCK, 2015] ROCK (2015). The robot construction kit ROCK web site. `http://rock-robotics.org` , *laste visited 02.03.2015*. [82, 86, 125]

[Sadhukhan and Moore, 2003] Sadhukhan, D. and Moore, C. A. (2003). Online terrain estimation using internal sensors. In *Proceedings of the Florida conference on recent advances in robotics*. [15, 18, 24]

[Sancho-pradel and Gao, 2010] Sancho-pradel, D. L. and Gao, Y. (2010). A survey on terrain assessment techniques for autonomous operation of planetary robots. *JBIS - Journal of the British Interplanetary Society*, 63(5-6):206 – 217. [2, 41]

[Schmid, 1995] Schmid, I. C. (1995). Interaction of vehicle and terrain results from 10 years research at {IKK}. *Journal of Terramechanics*, 32(1):3 – 26. [21]

[Scott and M., 2009] Scott, G. P. and M., S. C. (2009). The development of a soil trafficability model for legged vehicles on granular soils. *Journal of Terramechanics, Elsevier, June–August 2012*. [20]

[Silva and Tenreiro Machado, 2007] Silva, M. F. and Tenreiro Machado, J. (2007). A historical perspective of legged robots. *Journal of Vibration and Control*, 13(9-10):1447–1486. [1]

[Sonsalla et al., 2014] Sonsalla, R., Ahmed, M., Fritsche, M., Akpo, J. B., and Vögele, T. (2014). Scout rover applications for forward acqusition of soil and terrain data. In *Proceedings of European Planetary Science Congress (EPSC-2014)*, volume 9, Cascais, Portugal, Portugal. EPSC. [11]

[Spenneberg et al., 2005] Spenneberg, D., Albrecht, M., and Backhaus, T. (2005). M.o.n.s.t.e.r.: A new behavior-based microkernel for mobile robots. In *Proceedings of the ECMR 2005.* [33]

[Stavens et al., 2007] Stavens, D., Hoffmann, G., and Thrun, S. (2007). Online speed adaptation using supervised learning for high-speed, off-road autonomous driving. In *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, pages 2218–2224. [17, 18]

[Stelzer et al., 2012] Stelzer, A., Hirschmüller, H., and Görner, M. (2012). Stereo-vision-based navigation of a six-legged walking robot in unknown rough terrain,. *The*

International Journal of Robotics Research, Special Issue on Robot Vision, 31, Issue 4:381–402. []

[Tao et al., 2006] Tao, J. G., Wang, L., Deng, Z. Q., Gao, H. B., and Quan, Q. Q. (2006). Mechanical analysis and measurement of parameters of wheel -soil interaction for a lunar rover. *Journal of Physics: Conference Series*, 48(1):1222. [38]

[Terzaghi, 1943] Terzaghi, K. (1943). Theoretical soil mechanics. *3rd Ed., John Wiley and Sons, Inc., London*. [20, 41]

[TransTerrA Project, 2015] TransTerrA Project (2015). Semi-autonomous cooperative exploration of planetary surfaces including the installation of a logistic chain as well as consideration of the terrestrial applicability of individual aspects. `http://robotik.dfki-bremen.de/en/research/projects/transterra.html`, *laste visited 02.03.2015*. [11]

[Verruijt, 2012] Verruijt, A. (2012). *Soil Mechanics*. Delft Academic Press. [21]

[Virtual Crater Project, 2012] Virtual Crater Project (2012). `http://robotik.dfki-bremen.de/en/research/projects/virtual-crater.html`, *laste visited 02.03.2014*. [92]

[Weiss et al., 2006] Weiss, C., Fröhlich, H., and Zell, A. (2006). Vibration-based terrain classification using support vector machines. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4429–4434. [15, 16, 17, 18]

[Wong, 2014] Wong, J. (2014). Terramechanics and its applications to the evaluation of terrestrial and extraterrestrial vehicle mobility: theory into practice. *International Journal of Vehicle Design*, 65:384–410. [14, 37, 38, 39]

[Yeomans et al., 2013] Yeomans, B., Saaj, C. M., and Winnendael, M. V. (2013). Walking planetary rovers – experimental analysis and modelling of leg thrust in loose granular soils. *Journal of Terramechanics*, 50(2):107–120. [20]

[Yong et al., 1980] Yong, R., Boonsinsuk, P., and Fattah, E. (1980). Tyre flexibility and mobility on soft soils. *Journal of Terramechanics*, 17(1):43 – 58. [21]

[Yoo et al., 2010a] Yoo, Y.-H., Ahmed, M., Bartsch, S., and Kirchner, F. (2010a). Realistic simulation of extraterrestrial legged robot in trade-off between accuracy and simulation time. In *Proceeding of the 35th Annual Conference of the IEEE Industrial Electronics Society (IECON-2010)*, Glendale, AZ, USA. [11]

[Yoo et al., 2009] Yoo, Y.-H., Ahmed, M., Roemmermann, M., and Kirchner, F. (2009). A simulation-based design of extraterrestrial six-legged robot system. In *2009 35th Annual Conference of IEEE Industrial Electronics*, pages 2181–2186, Porto, Portugal. IEEE. [11]

[Yoo et al., 2010b] Yoo, Y.-H., Jung, T., Langosz, M., Rast, M., Kirchner, F., and Rossmann, J. (2010b). Developing a virtual environment for extraterrestrial legged robot with focus on lunar crater exploration. In *Proceeding of 10th International Symposium on Artificial Intelligent, Robotics and Automation in Space*, pages 206–213, Sapporo, Japan. [92]

[Yüksel et al., 2014] Yüksel, M., Ahmed, M., Girault, B., Birnschein, T., and Kirchner, F. (2014). A framework for design, test, and validation of electric car modules. In Fischer-Wolfarth, J. and Meyer, G., editors, *Advanced Microsystems for Automotive Applications 2014*, Lecture Notes in Mobility, pages 245–254. Springer International Publishing. [11]

[Yüksel et al., 2014] Yüksel, M., Oekermann, C., Girault, B., and Ahmed, M. (2014). Using industrial actuators for rapid development of electric car applications. In *Proceedings of the 14th International Conference on New Actuators (ACTUATOR-14); 8th International Exhibition on Smart Actuators and Drive Systems*, Bremen, Germany. MESSE BREMEN, WFB Wirtschaftsförderung Bremen , Bremen. [11]

[Zang and Zhao, 2013] Zang, M. and Zhao, C. (2013). Numerical simulation of rigid wheel running behavior on sand terrain. *APCOM & ISCM*. [21, 43]