



Politecnico di Bari

Repository Istituzionale dei Prodotti della Ricerca del Politecnico di Bari

Data-driven modelling and estimation of mechanical systems

This is a PhD Thesis

Original Citation:

Availability:

This version is available at <http://hdl.handle.net/11589/251820> since: 2023-04-20

Published version

<http://hdl.handle.net/11589/251820>
DOI: 10.6092/poliba/iris/vulpi-fabio_phd2023

Terms of use:

Altro tipo di accesso

(Article begins on next page)

19 April 2024



Department of Mechanics, Mathematics and Management
Ph.D. Program in Mechanical and Management
Engineering
SSD: ING-IND/13–APPLIED MECHANICS

Final Dissertation

Data-driven modelling and estimation of mechanical systems

by
Vulpi Fabio

Supervisors:

Prof. Reina Giulio

Dr. Milella Annalisa (CNR-STIIMA)

*Coordinator of Ph.D. Program:
Prof. Demelio Giuseppe Pompeo*

Course n°35, 01/11/2019-31/10/2022



Politecnico
di Bari

Department of Mechanics, Mathematics and Management
Ph.D. Program in Mechanical and Management
Engineering

SSD: ING-IND/13–APPLIED MECHANICS

Final Dissertation

Data-driven modelling and estimation of mechanical systems

by

Vulpi Fabio

Supervisors:

Prof. Reina Giulio

Dr. Milella Annalisa (CNR-STIIMA)

Coordinator of Ph.D. Program:

Prof. Demelio Giuseppe Pompeo

Acknowledgments

- Creating Machine Learning models and Artificial Intelligence is not about delegating thinking to a machine, but rather about teaching the machine to emulate your thinking and then continuously enhancing it. -

I am deeply grateful to Professor Giulio Reina and Senior Researcher Annalisa Milella who have provided invaluable guidance, encouragement, and support throughout the course of my research. Their expertise and mentorship has been instrumental in helping me achieve the goals of this thesis. They pushed me to meet their expectations and motivated me to overcome them, and for this reason I would like to apologise if I ever failed in doing so.

I am thankful to Dr. Antonio Petitti, Dr. Roberto Marani, Dr. Antonio Leanza and Dr. Vito Renò, who have patiently shared their knowledge and skills with me throughout the years of my research. Their support has been invaluable to me.

I would also like to express my gratitude to Senior Researcher Tiziana D’Orazio at CNR-STIIMA of Bari, who has provided me with the resources and opportunities to pursue my academic interests.

The financial support of the projects: Autonomous Decision making in very long traverses (ADE), H2020 (Grant no. 821988); Agricultural in TeroperabiLity and Analysis System (ATLAS), H2020 (Grant no. 857125); multimodal sensing for individual plANT phenOtypiNg in agriculture rObotics (ANTONIO), ICT-AGRI-FOOD COFUND (Grant no. 41946); “E-crops Technologies for Digital and Sustainable Agriculture” funded by the Italian Ministry of University and Research (MUR) under the PON Agrifood Program (No. ARS01_01136); CNR DIITET project “DIT.AD022.180 Transizione industriale e resilienza delle Società post-Covid19 (FOE 2020)”, sub task activity “Agro-Sensing”, is gratefully acknowledged.

The author is grateful to Cantina San Donaci agricultural farm for hosting experimental tests. The administrative and technical support by Michele Attolico and Giuseppe Bono is also gratefully acknowledged.

This thesis is dedicated to my family, who has always been my source of inspiration and motivation.

Abstract

This thesis examines the use of machine learning and deep learning in mechanical engineering problems. An underlying theme of the research is the comparison of Convolutional Neural Networks and Recurrent Neural Networks with standard machine learning techniques, such as Support Vector Machines, showing the advantages of Deep Learning in modeling complex phenomena where traditional approaches fall short. As an example, it is shown that with the proposed Multichannel Spectrograms, an autonomous robot can classify the traversed terrain with an accuracy of over 90% when using a CNN based on proprioceptive signals. This work addresses both generalization and extrapolation issues, demonstrating that Deep Learning delivers more robust results than standard machine learning when applied to data obtained in varied conditions. The importance of sensor data complementarity is also emphasized showing that the use of appearance-based measurements can be used to predict wheel-terrain interactions, improving the ability of planetary exploration rovers to avoid hazards and respond to changing conditions. The combination of proprioceptive and exteroceptive signals is explored for crop monitoring in agricultural robotics. The design and development of a multi-sensor system for 3D reconstruction of agricultural environments is here detailed together with the necessary sensor fusion algorithms. Additionally, a new unsupervised learning algorithm, Kalman Supervised Network, is proposed for modeling mechanical systems. KSN uses the principles of Kalman Filters to continuously learn from sensor measurements and produce a more accurate model of the system than the one embedded in the KF.

Contents

List of Figures	vii
List of Tables	xi
1 Introduction	1
1.1 Machine Learning and Deep Learning	1
1.2 Applications for Mechanical Systems	2
1.3 ADE Project	3
1.4 Rovers Overview	4
1.5 Proprioceptive Learning of Terrain	7
1.6 Exteroceptive prediction of vehicle-terrain interaction	9
1.7 Combined Perception for Crop Monitoring	10
1.8 State Estimation	11
2 Proprioceptive Learning for Classification	13
2.1 Learning Algorithms for Terrain Classification	13
2.2 Support Vector Machine	16
2.3 Convolutional Neural Networks	17
2.3.1 Multichannel Spectrograms	17
2.3.2 CNN Architecture	19
2.3.3 SherpaTT rover application	20
2.3.4 Experimental Results	22
2.4 Recurrent Neural Networks	25
2.4.1 RNN Architecture	26
2.4.2 Long-Short Term Memory RNNs	27
2.4.3 Convolutional LSTM	28
2.4.4 Husky rover application	29
2.4.5 Experimental Results	32
2.5 The role of feature and signal selection	40
2.5.1 Learning Parameters	42
2.5.2 SherpaTT rover application	43
2.5.3 Signal Augmentation	45
2.5.4 Feature Extraction	48

2.5.5	Experimental Results	53
3	Exteroceptive Learning for Regression	63
3.1	Motion Resistance Prediction	63
3.2	Husky rover application	64
3.3	Experimental Results	66
4	Exteroceptive and Proprioceptive signals integration	71
4.1	Crop Monitoring Application	71
4.2	Multi-view Mapping	73
4.2.1	Multi-sensor Device	74
4.2.2	Multi-view 3D Mapping	78
4.2.3	Simulation Environment	81
4.2.4	In-field Testing	82
5	Kalman Supervised Network	91
5.1	Problem Statement	91
5.2	KSN usage	93
5.3	Numerical Results	94
5.3.1	The mass-spring-damper system	94
5.3.2	KF Implementation	96
5.3.3	KSN Training	97
5.3.4	KSN Testing	98
6	Conclusions and future work	103
A	Author's Publications	105
	References	107

List of Figures

1.4.0.1 (a) SherpaTT in soft sand dunes during Morocco field trials in 2018; (b) SherpaTT in a sandy trench during the ADE final field tests in spring 2021.	4
1.4.0.2 Dummy figure	5
1.4.0.3 CNR-bot rover	6
1.4.0.4 Husky rover.	7
2.3.2.1 Architecture of the convolutional neural network.	20
2.3.3.1 Types of surfaces traversed by SherpaTT during the test and development of the system.	21
2.3.4.1 Confusion matrices obtained from (a) CNN trained with IMU data (b) CNN trained with Load Cell data (c) SVM trained with IMU data (d) SVM trained with Load Cell data	23
2.3.4.2 Confusion matrixes obtained from the CNN (a) and SVM (b) terrain classifier trained combining inertial and force signals.	24
2.4.1.1 Recurrent Neural Network structure. In this study $X_t \in \mathbf{R}^{10}, Y_t \in \mathbf{R}^{15}$	26
2.4.2.1 Long Short-Term Memory Model Structure. The number of channels is nCh , the number of hidden units is Nh and the number of terrain classes is nCl	28
2.4.3.1 Convoluted Long Short-Term Memory Model Structure. The number of channels is nCh , the number of filters is $nFilt$, the number of hidden units is Nh and the number of terrain classes is nCl	29
2.4.4.1 Pipeline of signal partition and sample S_i extraction, with PW the time window used for partitioning, MW the time window used for augmentation and ST the corresponding stride.	31
2.4.4.2 CNN input sample for $nCh = 4, MW = 1.5s, wL = 0.4s$ and $wO = 0.2s$	32
2.4.5.1 Accuracy of respectively LSTM (blue), C-LSTM (red), SVM (green) and CNN (yellow) models for each sample length of terrain tested corresponding to different SL values. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)	33
2.4.5.2 Sensitivity of the LSTM, C-LSTM, CNN and SVM models for each MW tested.	34

2.4.5.3	Precision of respectively LSTM, C-LSTM, CNN and SVM models for each MW tested.	35
2.4.5.4	F1-score of respectively LSTM, C-LSTM, CNN and SVM models for each MW tested.	36
2.4.5.5	Normalized Confusion Matrixes for all models and $MW = 1.7$ (SL = 85 cm).	37
2.4.5.6	Average classification time as a function of the sample length LSTM LSTM (blue), C-LSTM (red), SVM (green) and CNN (yellow) models for each sample length of terrain tested corresponding to different SL values.	38
2.5.2.1	Vertical acceleration and drive torque (wheel front left) measured while SherpaTT driving straight on different terrains.	44
2.5.3.1	Definition of vertical force offset (dx).	47
2.5.3.2	Torque applied by the left front wheel of SherpaTT as obtained from direct measurement of the load cell (solid grey line), indirect measurement via the electric current drawn by the motor (solid black line), or alternatively via the longitudinal force provided by the load cell (dashed black line).	47
2.5.4.1	PC an WB indices distribution for the most relevant features.	50
2.5.4.2	Block diagram of the proposed feature selection algorithm.	51
2.5.4.3	PC an WB indices distribution for the most relevant features.	52
2.5.5.1	Generalization results for best features SVM and best signals CNN. CNN, convolutional neural network; SVM, support vector machine. . .	55
2.5.5.2	Extrapolation results for best features SVM and corresponding signals CNN.	57
2.5.5.3	(a) Sherpa TT during the sand mine testing; (b) a close up of the tracks left by the wheels.	58
2.5.5.4	Semantic labeling using discrete terrain difficulty categories: (a) 3D stereo-generated map of the environment with overlaid the path (dashed white line) followed by the rover, (b) corresponding terrain difficulty visualization. Terrain patches are marked respectively in red, yellow, and green, for high, medium, and low difficulty.	59
3.2.0.1	Robotic platform used for data gathering.	65
3.3.0.1	Terrain types: (a)-(b) ploughed agricultural terrain; (c)-(d) compact agricultural terrain. Left: sample images with overlaid the boundaries of the inspection window. Right: corresponding 3D colour terrain patches obtained from the stereocamera.	68
3.3.0.2	LSBoost predictions (a) for ploughed terrain and (b) for compact terrain.	69
3.3.0.3	LSTM-RNN predictions (a) for ploughed terrain and (b) for compact terrain.	70

4.2.0.1 (a) Demonstrator of the UGV's sensor box: (1)-(2) Intel NUC Windows PCs; (3) Batteries, (4) T265 Camera, (5) D435 Cameras, (6) X-Sense MTI-300 IMU, (7) U-blox ZED F9P board, (8) Sensor Box GPS antenna. (b) Closeup of the multi-camera system. (c) CAD model of the sensor frame.	74
4.2.1.1 Sensor box data flow.	75
4.2.1.2 Schematic of the image acquisition software.	76
4.2.1.3 User interface of the multi-view camera system: (a) interface for data acquisition and storage (MultiBagWriter); (b) interface for reading stored image databases (MultiBagReader).	77
4.2.3.1 Gazebo simulation environment composed of a vineyard row and a mobile robot.	81
4.2.4.1 Google Earth view of the four paths estimated by EIF in a commercial vineyard, San Donaci, Apulia Region, Italy ($40^{\circ}27'16.2''N$ $17^{\circ}54'30.6''E$).	83
4.2.4.2 Localization results for Test 1: (a) from RTK-GPS only; (b) from T265 camera only; (c) after EIF fusion of RTK-GPS, IMU and T265 measurements. In (a), red diamonds are overlaid in two different zones without RTK coverage due to connection loss.	85
4.2.4.3 Localization results (a) for a second run along the same path of Test 2 (b) for Test 3 (c) for Test 4; RTK-GPS only (solid grey line), T265 only (dashed black line) and EIF (solid black line). Start and stop positions for EIF trajectory are denoted by green and red dot, respectively.	86
4.2.4.4 EIF-derived path overlaid on Google Earth view for Test 1. Three successive positions of the robot are pinpointed. For these positions, the corresponding visual data are shown in Figure 4.2.4.5.	87
4.2.4.5 Output of the multi-view camera system for three robot locations along the path (Test 1): (first row) color images, (second row) depth images obtained from IR stereo reconstruction and (third row) multi-view 3D point cloud.	87
4.2.4.6 Mapping results (Test 1): upper view of the terrain map reconstructed by the central camera. The robot trajectory estimated by the sensor fusion approach is also overlaid.	88
4.2.4.7 Mapping results (Test 1): closeup of loop closure before (a) and after (b) EIF correction.	88
4.2.4.8 Closeup of the multi-view map for Test 1 (first 20 m): (a) RGB, (b) GRVI and (c) elevation map. In (b), green points refer to vegetation, whereas blue points correspond to non-vegetated parts. Lighter green denotes higher GRVI values. In (c), a jet colormap is used to represent point height with respect to ground.	89
5.1.0.1 example	92

5.2.0.1 (a) Block diagram representing the training stage of the Kalman Supervised Network (b) block diagram representing the proposed estimation scheme. Time \bar{t} marks the beginning of the measurement outage when predictions of KSN are propagated alongside KF embedded physical model predictions. \tilde{x}_t refers to the network prediction at time $t > \bar{t}$. . .	94
5.3.1.1 Mass-spring-damper system, where m is the mass of the system, k is the elastic coefficient of the spring, c is the damping coefficient, $h(t)$ is the input and $x(t)$ is the mass displacement	95
5.3.3.1 Kalman Supervised Network layout used for the considered mechanical system	97
5.3.4.1 Testing results obtained by comparing state estimation as obtained from the KSN and the standard KF during measurement outage starting at time $t = 3$ s	100

List of Tables

2.4.4.1	Considered Terrain Types and number of available 5s long windows for each one	30
2.4.5.1	Architecture parameters of DL algorithms.	32
2.4.5.2	Mean and standard deviation for accuracy values corresponding to the MW range $[1.3s, 2s]$ (i.e. $[65cm, 1m]$).	35
2.4.5.3	Model’s computation burden for the classification of a single observation. For all models it is assumed $MW = 1.7s$ (i.e., $SL = 85cm$).	39
2.5.1.1	Learning parameters of SVM Classifier.	42
2.5.1.2	Learning parameters and Architecture parameters of CNN.	42
2.5.2.1	List of available proprioceptive signals.	43
2.5.3.1	List of indirect signals.	46
2.5.4.1	List of parameters involved in the feature selection approach	52
2.5.4.2	Best feature set	53
2.5.5.1	Performance comparison between terrain classifiers trained on different feature sets: direct, full, best feature set	55
2.5.5.2	Accuracy, Precision, Recall and F1-score for SVM and CNN in generalization	55
2.5.5.3	Precision, Recall and F1-score for SVM and CNN in extrapolation using varying velocity	57
2.5.5.4	Category of difficulty assigned to each terrain type of the training set and predictions as obtained from SVM and CNN in the sand mine test	58
3.2.0.1	Hyper-parameters of the LSBoost model.	65
3.3.0.1	Optimized LSBoost ensemble hyper-parameters.	66
3.3.0.2	Hyper-parameters of the LSTM-RNN model.	67
3.3.0.3	Model errors.	67
4.2.4.1	Comparison of different localization sources along four robot paths (Test 1 to 4): discrepancy in the East-North-Up frame between the starting and ending points of the trajectory expressed in terms of 3D Euclidean distance (D), 2D Euclidean distance in the motion plane (D_{EN}), altitude distance (D_U), and standard deviation of altitude measurements (σ_U) along the entire path.	83

5.3.1.1 True system parameters	96
5.3.1.2 True noise variances	97

Chapter 1

Introduction

1.1 Machine Learning and Deep Learning

Machine learning is a branch of artificial intelligence that allows systems to automatically improve performance through experience, without being explicitly programmed. It involves training algorithms on data to identify patterns and make predictions or decisions in a specific task. The main types of machine learning algorithms are:

- Supervised learning - algorithms are trained on labeled data to make predictions on new data. Examples: linear regression for predicting numerical values, classification algorithms for identifying objects in images.
- Unsupervised learning - algorithms learn from unlabeled data to identify patterns or groupings. Examples: clustering algorithms for grouping similar items in a dataset, dimensionality reduction for visualizing high-dimensional data.
- Reinforcement learning - algorithms learn from interactions with an environment to maximize a reward signal. Examples: control algorithms for optimizing process control in manufacturing, robotics algorithms for autonomous vehicles.

Deep learning, a subfield of machine learning, uses artificial neural networks with multiple layers to solve complex problems. It is capable of automatically learning features and representations from raw data, making it well suited for tasks such as image and speech recognition, and natural language processing. Examples of deep learning algorithms include Convolutional Neural Networks (CNN) for image classification in computer vision, Recurrent Neural Networks (RNN) for sequential data processing in speech recognition, and Generative Adversarial Networks (GAN) for synthesizing images in computer graphics. Machine learning has numerous applications in the engineering field such as predictive maintenance for industrial equipment, autonomous control for smart buildings, and quality control in manufacturing. Deep learning is used in various applications such as object detection and classification in computer vision, speech recognition in human-computer interaction, and natural language processing in language translation.

1.2 Applications for Mechanical Systems

Machine learning algorithms and deep learning algorithms have numerous potential applications in the mechanical engineering field. Here are a few examples with references to existing literatures:

- Predictive maintenance in mechanical systems: Machine learning algorithms can be used to predict the likelihood of failure in mechanical systems [20, 70].
- Quality control in manufacturing: Machine learning algorithms can be used to detect and classify defects in manufactured products [100, 98].
- Autonomous control in smart buildings: Machine learning algorithms can be used to optimize heating, ventilation, and air conditioning (HVAC) systems in smart buildings [136, 2].
- Fluid dynamics simulations: Deep learning algorithms can be used to speed up fluid dynamics simulations [68, 85].
- Online estimation of parameters: Machine learning algorithms can be used to estimate parameters in real-time. For example the angle of internal friction [137] or soil cohesion [76].
- Modeling of materials: Machine learning algorithms can be used to model the behavior of materials, such as the prediction of the Curie temperature [90] or vibration free energy and entropy using solely the chemical composition of the material [69].
- Prediction of system dynamics: Machine learning algorithms can be used to predict the dynamics of complex systems. Building data-driven models for longitudinal and lateral dynamics [133] or for estimating the slip ratio [134].

These are just a few examples of the many potential applications of machine learning and deep learning algorithms in the mechanical engineering field. There is a growing body of research exploring the use of these algorithms in various areas of mechanical engineering, and the field is likely to continue to grow in the coming years.

Machine learning models are built using data, which can be proprioceptive or exteroceptive. Proprioceptive data refers to information generated by sensors within the system being modeled, while exteroceptive data refers to information generated by sensors outside the system. The distinction between sensing and perception is important in understanding the role of data in machine learning models. Sensing refers to the process of gathering information about the environment through the use of sensors. Perception refers to the process of interpreting and making sense of that information. In the context of machine learning, proprioceptive perception refers to the process of making sense of proprioceptive data, while exteroceptive perception refers to the process of making sense of exteroceptive data. Proprioceptive data is often used to build models that reflect the

internal state of a system, while exteroceptive data is often used to build models that reflect the external state of a system. For example, in the case of a self-driving car, proprioceptive data might include information about the vehicle's speed, steering angle, and other internal state variables, while exteroceptive data might include information about the road, other vehicles, and the surrounding environment. Machine learning algorithms can be used to build models that use both proprioceptive and exteroceptive data to make decisions and control the vehicle.

Chapter 2 of this thesis focuses on deep learning applications for terrain recognition using solely proprioceptive data. Chapter 3 presents a novel approach for exteroceptive measures to predict vehicle-terrain interaction. Finally, Chapter 4 describes the process of combining exteroceptive and proprioceptive measures and section 5 presents a novel approach for unsupervised accurate modelling of generic mechanical system based on the Kalman Filter algorithm.

1.3 ADE Project

The study of this thesis has been developed as part of the research activity for the project ADE (Autonomous DEcision making in very long traverses) [91, 37]. ADE is an ongoing European H2020 research project, part of the PERASPERA second call [38]. Its ambition is to design, develop, and test in a representative Earth analogue a fully autonomous rover system. The ADE system is capable to take all decisions to pursue mission objectives, to increase data collection and overall science, to perform autonomous long traverse surface exploration, to guarantee fast reaction and adapt to unforeseen situations, increasing mission reliability, and guaranteeing optimal exploitation of resources. The main goal of ADE is to develop and test a rover system capable to achieve autonomous long-range navigation in hostile environments while guaranteeing fast reaction, mission reliability and safety, and optimal exploitation of the robot's resources within reasonable costs.

Future generations of mobile robots will be required to explore areas, which present highly challenging mobility conditions. In order to fulfill long distance and duration missions, it will be important to be able to understand the type of traversed surface, so that adequate control and planning strategies can be implemented and areas of high risk can be properly negotiated or avoided. This thesis details one of the key technologies to enable long range applications of planetary rovers related with terrain awareness. As a matter of fact, the mobility range capability of the rovers has been up to date strongly limited to few tens of meters per sol day [89, 36, 88]. From a purely technical point of view, this limitation has both hardware and software sources. The former and most important is the finite power storage of the rover locomotion system, which is fixed given a robot design. The latter is reduced skills in terms of autonomous decision-making. The result is the impossibility to cover reasonable portions of/or multiples geographical areas of a potential planetary surface, reducing drastically the data returns both in terms of "pure science" and/or potential data collection for in-situ resources analysis and further exploitation. The capability of taking autonomous decisions on-board can be improved

by artificial intelligence. Improving it will extend the autonomy of the rover across multiple geographical areas and therefore expands opportunities of data collection.

The importance of sensing hazards in long-range navigation was highlighted in April 2005, when the Mars Exploration Rover Opportunity became embedded in a dune of loosely packed drift material [28]. The terrain geometry as reconstructed from a distance via stereovision did not indicate any hazard, however, the high compressibility of the loose drift material caused the wheels to sink deeply into the surface. The combination of the drift's low internal friction and the motion resistance due to sinkage prevented the rover from producing sufficient thrust to travel up the slope. Opportunity's progress was delayed for more than a month while engineers worked to extricate it. A similar embedding event experienced by the Spirit rover in 2010 led to the end of its mobility operations [51]. This highlights the need for future generations of planetary exploration rovers to have key technologies that can overcome these limitations, performing long traverses while guaranteeing fast reaction, mission reliability and safety, and optimal exploitation of the robot's resources within reasonable costs. In this context, the ability to sense and characterize the incoming terrain would represent an enabling technology towards long-term autonomy and potential hazard avoidance [86].

1.4 Rovers Overview

SherpaTT SherpaTT was built by DFKI RIC for long-distance exploration applications [26], negotiation of highly challenging terrains or non-nominal conditions (sinkage in soft soil, getting entangled between rocks or alike), cooperation tasks between heterogeneous rovers in a collaborative sample return mission, search and rescue and/or security missions. SherpaTT is a four-wheeled mobile robot [25] outfitted with an actively articulated suspension system, independent drive and steer wheel motors and a six degrees of freedom (DOF) manipulation arm.



Figure 1.4.0.1: (a) SherpaTT in soft sand dunes during Morocco field trials in 2018; (b) SherpaTT in a sandy trench during the ADE final field tests in spring 2021.

Hence, the rover is a hybrid wheeled-leg rover, meaning it can take advantage of both, wheeled and legged locomotion according to the terrain difficulty. SherpaTT has a mass of about 180 kg and a payload capacity of at least 40 kg. Each of the four suspended legs has five DOF that include the rotation of the whole leg about the shoulder or pan axis with respect to the robot body, the two rotations of the inner and outer leg parallelograms for lifting a wheel, and the steer and drive angle of the wheel. Each of the 20 suspension and six arm joints delivers telemetry data at a rate of 100 Hz. The telemetry includes joint position (absolute and incremental), speed, current, PWM duty cycle and two temperatures (housing and motor windings). Additionally, a 6-DOF force-torque sensor (FTS) is placed at the mounting flange of each wheel-drive actuator enabling the direct measurement of the generalized forces that each wheel exchanges with the supporting surface. Active force control for the wheel-ground contact as well as the roll-pitch adaption are two processes of the so-called Ground Adaption Process (GAP) in SherpaTT.



Figure 1.4.0.2: Dummy figure

Polibot The polibot rover is a skid-steering trucked rover developed by Polytechnic of Bari that serves as a base for an expandible sensorial set-up. It can be equipped with RGB-D cameras, stereo cameras, LiDARs, more than one RTK-GPS antenna for enhanced heading estimation (as in 1.4.0.2) and IMUs. Chapter 3 is used for the crop monitoring purpose and equipped with the SensorBox as shown in Figure 1.4.0.2 (a).

CNR-bot The CNR-bot is a custom-built robotic platform available at the Mobile Robotics Laboratory of CNR-STIIMA, Bari, for crop monitoring purposes. CNR-bot is equipped with four independent steering wheels that provide exceptional maneuverability, allowing for a wide range of motion in all directions and also in-place rotation. The four wheels can be driven independently, providing maximum control and versatility. The ability to steer each wheel independently enables the rover to navigate obstacles, perform complex movements, and traverse difficult terrains.

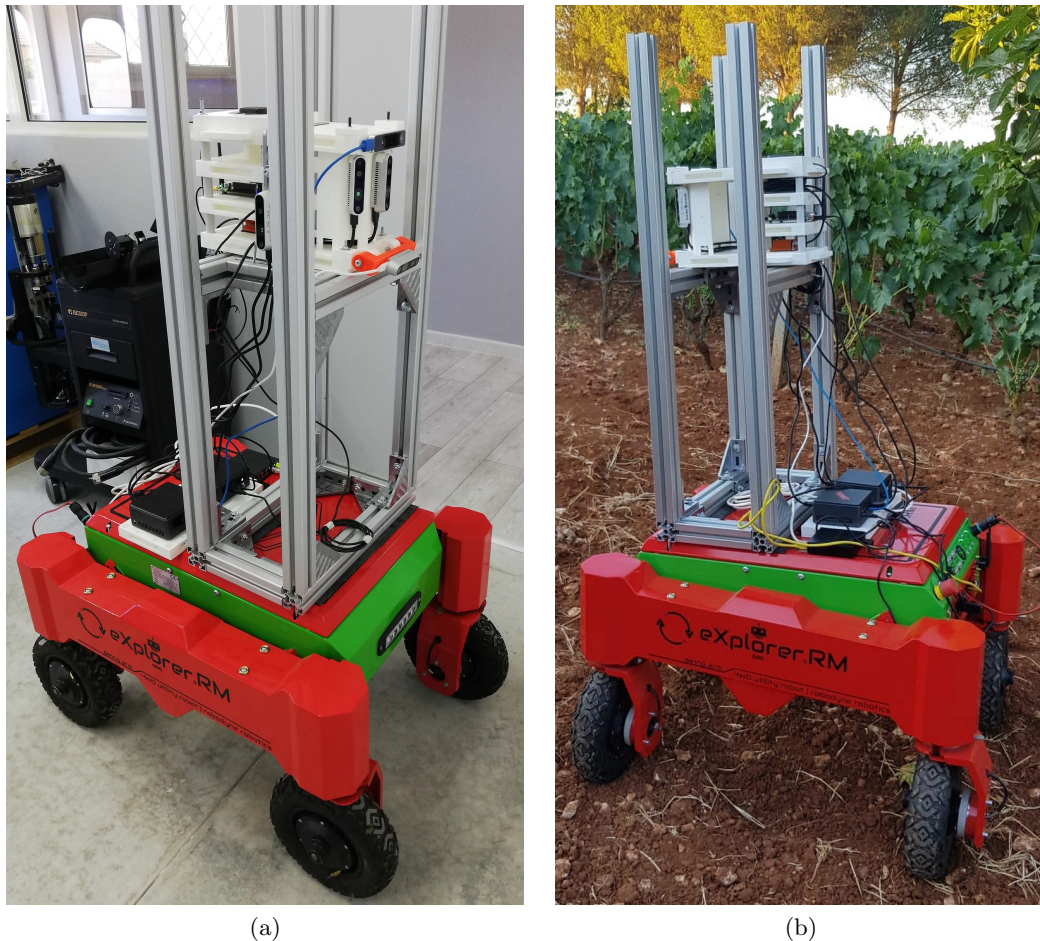


Figure 1.4.0.3: CNR-bot rover

Husky The rover Husky has a length of 0.7 m and a width of 0.5 m, and its proprioceptive sensor suite is composed of encoders to measure wheel angular velocity, electrical current sensors that provide an indirect measurement of tire torque, and an XSENS MTi-300 inertial sensor module that tracks linear accelerations and angular rates. The sensor suite is completed by an exteroceptive counterpart that includes a stereo-camera, an outdoor laser rangefinder, and a GPS



Figure 1.4.0.4: Husky rover.

1.5 Proprioceptive Learning of Terrain

The envisaged idea presented in this paper is that terrain properties can be obtained by the rover's wheels that serve as tactile sensors. SherpaTT's wheels are outfitted with six axis load cells that provide a direct measurement of the forces and torques applied by the wheels to the ground and vice versa. In addition, estimation of the motion states in terms of accelerations and rate-of-turn can be gathered by an inertial measurement unit attached to the robot's body. All 20 joints of the suspension system provide telemetry including currents, voltages, PWM, temperatures, velocity, and position. Following this rationale, signals that are directly modulated by the terrain interaction can be obtained. These measurements represent a rich source of information that bring significant content in terms of terrain properties. The work presented in this paper applies learning approaches to this data in order to make intelligent autonomous robots adaptive to the

site-specific environment [32], [111].

In this study we generally refer to terrain classification as the task of recognizing from among a list of known possible types the one crossed (or to be crossed) based on the analysis of the sensory data available onboard the robot [129]. This problem is common to different application areas. Notable examples can be found in robotics for surface planetary exploration as described in [6, 48, 96, 46], and precision agriculture for which terrain identification plays a key role in the fulfillment of numerous tasks such as seeding, ploughing, fertilizing or controlled traffic [111, 87]. Early research relied on forward imaging sensing and used limited learning [43]. The visual appearance of distant terrain has been used in [105] also combining distance measurements generated by stereovision [110], radar [82] and lidar [117]. However, observation of a given terrain from a distance does not provide any information about its mechanical properties that directly impact on vehicle mobility. It is known that off-road traversability largely depends on the interaction between the robot and the terrain [10]. Dynamic ill-effects including wheel sinkage, slippage and rolling resistance are the result of this complex interplay. For example, the ground can be considered drivable based on the geometric elevation map. Yet, the robot can incur in serious risks if this terrain offers low traction properties due to high slippage and consequent lack of progression, as explained in [4].

Therefore, recently, methods that use proprioceptive sensing have been also proposed for terrain classification [18], [123, 73, 42, 92]. The envisaged idea behind this approach is that, from a mechanical perspective, terrain category can be identified using wheels as tactile sensors that generate signals modulated by the vehicle-terrain interaction. Hence, proprioceptive data contain much information, which can be useful to characterize the terrain type. In addition, learning-based approaches have been introduced in order to make intelligent autonomous robots adaptive to the sitespecific environment [13, 47, 79, 80]. One of the objectives of this study is to demonstrate the potential of terrain classification via learning algorithms that are trained on proprioceptive features. Here, proprioceptive features refer to statistics that are extracted from the measurement of a physical variable pertaining to the robot-environment interaction, for example, wheel velocity, forces, body linear, and angular accelerations. The hypothesis is that being modulated by the terrain properties, these features are a rich source of information from which the specific terrain type can be inferred via learning approaches [18, 45].

More difficult the environment, less likely expert rule-based or heuristic strategies perform well. This is the case for natural terrains that entail many challenges including variability in surface and lighting conditions, lack of structure, no prior information, and in which learning approaches may fit better. Therefore, information pertaining to wheel-terrain interaction can be extracted and, then, a mapping between proprioceptive data and the corresponding surface can be created. A learning approach fits well to this application as: i) the large number of parameters involved make a physics-based terrain model rather complex, ii) the mapping from proprioceptive input to a given terrain's mechanical properties is an extremely complicated function, for which a closed analytical form is very difficult to obtain and one possible solution is to observe the phenomenon and learn about it through training examples, iii) adaptability of the ve-

hicle's behavior can be promoted through learning. Following recent research trends, this paper tackles the terrain classification problem for rough terrain vehicles relying on proprioceptive sensing and deep learning. Two types of network are discussed here: the Recurrent Neural Network (RNN) and the spectrogram-based Convolutional Neural Network (CNN). RNNs can be further engineered with different structures. This work focuses on Long Short-Term Memory recurrent neural network (LSTM) and Convolutional Long Short-Term Memory recurrent neural network (C-LSTM). The performance of LSTM, C-LSTM and CNN are evaluated in comparison with the well-known state-of-the-art machine learning classifier Support Vector Machine (SVM), showing similar or improved results even with relatively similar terrains. Our hypothesis is that self-learned features from deep learning may include temporal information from the data that are not captured by the manually designed features used by SVM.

1.6 Exteroceptive prediction of vehicle-terrain interaction

On natural terrain, as in agricultural and off-road settings, the mobility of a vehicle is known to be highly influenced by wheel-terrain interaction and can be very different on ploughed terrain rather than on dirt road or compacted soil [10]. The knowledge of the terrain characteristics and its ability to support vehicular motion can be beneficial in many ways. Locomotion performance can be optimised in terms of traction or power consumption (e.g., fuel or battery life) by adapting control and planning strategy to site-specific environments. Terrain estimation can also contribute to increase the safety of agricultural vehicles during operations near ditches, on hillsides and cross slopes, as well as on highly-deformable terrain [128]. Another important aspect that is raising interest in precision agriculture is related to the prediction of the risk of soil compaction by farm machinery [118]. Early work on vehicle-terrain interaction estimation mainly relied on model or expert rule-based observers [107, 93]. More recently machine learning approaches have been developed as an alternative or complementary solution with a focus on slip estimation [5, 111]. This thesis presents a novel framework for learning and predicting online the motion resistance of an agricultural robot on natural terrain. The main novelty of the proposed approach relies on the use of visual information, including appearance and geometric characteristics of the ground, to learn and predict from a distance the resistance torque encountered by the vehicle when traversing different terrain types. Estimation of the expected torque before entering a terrain is crucial for the vehicle to better plan its safest path and avoid dangerous or difficult areas. To address this problem, terrain appearance and geometry information are correlated to the torque measured by the rover while traversing a given surface. Such relationship is learned from previous experience (learning phase), so that, successively, motion resistance can be predicted from visual information only (prediction phase).

1.7 Combined Perception for Crop Monitoring

Persistent and timely crop monitoring is crucial for the development of sustainable food production systems. While remote sensing from satellites and aircrafts has been successfully used for some decades to allow for rapidly mapping and characterize wide areas through acquisition of multispectral imagery and 3D data, they generally lack the spatio-temporal resolution needed for precision agriculture or phenotyping tasks. In crops with smaller extension, Unmanned Aerial Vehicles (UAVs) have been effectively adopted for remote crop survey with higher spatial and temporal resolution compared to satellite and airborne devices. In high-density crops, however, collecting information on biophysical properties at plant or leaf/fruit level via aerial sensing may be still infeasible. As a result, ground-based sensing through Unmanned Ground Vehicles (UGVs) has been proposed for in-field close-range applications [106]. In this respect, imaging sensors embedded on ground vehicles have recently attracted much attention as an effective solution to collect high resolution proximal data and provide information on plant characteristics at centimetre or sub-centimetre scale. At the same time, they can be used for vehicle guidance automation and scene perception and understanding in general, thus contributing to all aspects of crop monitoring automation, from data gathering up to high-level data processing and interpretation. Among imaging sensors, consumer-grade RGB-D cameras, i.e., sensors that embed color and depth sensing into a unique device, have emerged in the last years in mobile robotics applications, to provide the vehicle with a real-time representation of both appearance and 3D structure of the environment. One shortcoming of typical consumergrade color-depth sensors is their limited field-of-view, which makes them unsuitable for applications that need continuous survey of extensive areas, like in agricultural settings.

In this thesis, a multi-view RGB-D camera system is proposed to enhance the perception ability of an unmanned agricultural robot. The system features three RGB-D cameras arranged to cover a horizontal field of view of about 130 deg. The cameras are integrated with localization sensors, including a stereo-based tracking camera, an RTK-GPS and an IMU, which provide accurate vehicle position information for image geo-referencing based on Extended Information Filter (EIF). All the sensors are physically integrated in a custom-built sensor box, which is designed to be self-contained from both a computational and energy point of view and independent from the particular vehicle architecture. The system is intended to generate accurate maps to facilitate operations on a narrow scale with a smaller environment footprint. To this end, a point cloud assembler that uses EIF-based pose estimates and the known relative poses between sensors is developed to reconstruct a geo-referenced multi-view 3-D map of the traversed environment, which could provide a useful input to precision farming technologies and crop monitoring tasks. An additional use of the map is that it can be incorporated into a high-fidelity simulator that can support the development and pre-testing of algorithms for autonomous driving. In this respect, the map can be converted into a mesh representation using the Ball Pivoting Algorithm (BPA) and imported into a simulation environment under Gazebo [40].

1.8 State Estimation

The estimation of the state is fundamental when dealing with the control of a dynamical system. In this respect, it is well known that if the system is linear with additive white Gaussian noise, then, the Kalman Filter (KF) is the optimal estimator that minimizes the Mean-Squared Error (MSE) [64] [58]. One of the advantages of the KF is the low-complexity: it is an iterative algorithm composed of two phases, i.e., predict and update. In the predict phase, the filter computes an a priori estimate by means of a model of the system. Then, in the update phase, the a posteriori estimate is computed thanks to the available measurements. When sensor signal is unavailable, as for example when dealing with long-term predictions, observations are not possible and the KF is only able to propagate predictions exploiting the physics based model, with a consequent accuracy reduction in state assessment [116].

To face this issue, in this work we introduce a hybrid data-driven system in which KF estimates are used to train a Neural Network (NN). The goal of the NN is to compute more reliable long-term predictions of the state without exploiting measurements and without training on the inaccessible groundtruth underlying dynamics. In the past decade, NNs have been widely applied in several application contexts [72, 9]. Neural Networks are data-driven non-linear black-box structures able to characterize the behavior of complex processes learning it from the data involved rather than fitting a formally defined model. For this reason, NNs are also a suitable tool when dealing with state estimation [63]. In [115], a KF-like data-driven recursive filter, called KalmanNet, is introduced. The aim of KalmanNet is to estimate the state of a dynamical system without the complete knowledge of the model. However, this methodology relies on the knowledge of the ground truth in the training phase. Moreover, in [1] a KF-based observer is proposed for the estimation of the tire lateral force and road grip potential. The observer is a hybrid system composed by an Extended KF (EKF), a recursive least square block, and a NN. The NN is used to identify the highly nonlinear behavior of the tire. This solution requires a large amount of labelled data collected with aggressive maneuvers, with which networks with known friction coefficients are trained. In [66], the estimation of the position of an autonomous car is investigated. In this case, a Deep Neural Network is applied to compensate for the weakness of each available sensor.

However, future values are needed for the training phase, leading to an extrapolation process. In this thesis the use of a NN trained on Kalman filter estimates is proposed. The aim is to learn the underlying dynamics from the corrections provided by the filter that exploits online measurements.

In this work we assume that the system dynamics is described by an unknown transfer function of a general complexity and with time independent parameters. This unknown transfer function is modelled by means of a physical description and the parameters are assessed with a certain degree of accuracy, like in a realistic scenario. A Kalman filter provides a posteriori estimates on the state of the observed system that are accurate (i.e. close to the underlying unknown transfer function), but generally not linked by a single function and instead produced with a loop of predictions and updates. On

the contrary, the physical model embedded in the Kalman filter uses always the same function to provide a priori predictions that are however further from the real dynamics than the filter estimates. The proposed net is used to learn the underlying dynamics of the considered system using KF observations as training examples. With this strategy we are able to produce a transfer function that provides more accurate predictions than the best available physical model embedded in the Kalman filter, incorporating the knowledge provided by the measurements to grasp the unknown underlying dynamics. The proposed KSN can be trained with relatively few and easy-to-collect data and does not require further knowledge of the system that is not typically available during field tests.

Chapter 2

Terramechanics Application

2.1 Learning Algorithms for Terrain Classification

Robotic mobility takes advantage of terrain classification by predicting interaction with the soil when optimizing controls or planning paths. Solving terrain-related challenges such as soil identification is an important research area in autonomous robots, alongside trajectory planning, localization, and obstacle avoidance [86]. The latest developments in terrain classification strategies show that researchers have been focusing on two main categories: visual (or exteroceptive) and visual-independent (or proprioceptive) methods. In both approaches, data collected from sensors are used to train machine or deep learning-based classifiers that enable identification of the traversed terrain.

The interesting problem of terrain characterization was addressed by terramechanics in [97] where a particle filter algorithm is coupled with regression analysis leading to optimal terramechanics parameters predictions of single wheel experimental measures in laboratory conditions. Cohesion, angle of internal friction and shear modulus, among other descriptive features are defined with a mathematical model to characterize the terrain and estimated through Bayesian techniques from measurable quantities such as drawbar pull, wheel input torque and sinkage.

In [111] the average values of motion resistance and slippage alongside root mean square and standard deviation of vertical acceleration were combined in a single four-element vector used as input for a proprioceptive support vector machine-based classifier. Results presented in [111] highlight the complementarity of proprioceptive and exteroceptive data especially to distinguish terrains with similar colors. The sensors used for visual perception include RGB cameras [119, 132], RGB-D cameras [79], LiDARs [119], visual cameras [95] and monocular cameras [7].

Although visual-based approaches are more common than proprioceptive-based ones, they have limitations as well. Therefore, researchers have investigated methods that use proprioceptive sensing for terrain classification. In this case, the sensors used to perceive the incoming terrain include inertial measurement unit (IMU), force-torque sensors, microphones, and wheel encoders. As an example, [61] used microphones to support an RGB camera in the dark conditions. Researchers in [18] measured vibrations

via accelerometers, analyzed them in the frequency domain and implemented an online classifier that relies on principal component analysis (PCA) for feature reduction.

Many feature extraction algorithms used for proprioceptive-based terrain classification use Fast Fourier Transform (FFT), Discrete Fourier Transform (DFT) or Power Spectral Density (PSD). A single-wheel testbed provided with a one-axis vibration sensor normal to the ground was used in [18] to collect data that are associated through FFT to feature vectors fed to an SVM model for terrain classification. The proprioceptive classifier proposed in [18] was also used to provide training examples to a visual terrain classifier following a self-supervised approach [19].

Optimal results and clear methods presented in [18] inspired researchers as in [6, 130, 131] where the three axes vibrations are first transformed with FFT and then concatenated. The feature vector containing FFT results is then used to construct an SVM model in [131] and a multilayer perception deep neural network in [6].

Reference [138] integrated the information provided by a 3-axis accelerometer, a single axis vibration sensor and one microphone through a multiclassifier combination principle. Different machine learning methods were investigated for terrain classification in [138], namely k-Nearest Neighbors (kNN), Naive Bayes (NB), SVM and Random Forest (RF), and different feature extraction algorithms are also tested such as Modified Mel-Frequency Cepstral Coefficient (MMFCC), FFT, Zero Crossing Rate (ZCR), Short Time Energy (STE), entropy, spectral centroid, spectral roll-off, and spectral flux.

Results presented in [138] suggest SVM is better suited for online terrain classification compared to the other tested algorithms. Accuracy values for SVM in [138] are between 80% and 90% for signals collected at relatively low speeds (0.4 m/s) depending on the number of features selected.

According to [6, 130, 131] and [138], rover's speed has a significant influence on proprioceptive-based terrain classification accuracy. Lower travelling speed corresponds to lower accuracy values because the signal-to-noise ratio is lower and significant frequency features contained in signals become undetectable by machine learning algorithms. Reaching good classification performance at low speed values is of great importance for proprioceptive-based terrain classification models, since it might be dangerous to travel at high speed on a terrain that is possibly unknown and that the rover is trying to identify.

The above works based on visual-independent approaches represent a step forward in the direction of providing a mobile robot with information about the mechanical properties of the terrain. Although they achieved high confidence levels, little effort was spent on feature selection as a means to reduce the computational burden of the model without penalties in performance. A reduction in the number of features used to train and test a machine learning classifier would lead to a lighter computational burden in terms of feature extraction time, testing time, and memory usage. One of the contributions of this research is to develop a feature selection algorithm demonstrating that these benefits can be achieved without compromising the accuracy of the model. A body of research has been devoted to the feature extraction process, as the quality of the feature space directly affects the accuracy of the associated classifier. The feature extraction strategy

depends on the machine learning approach chosen for terrain classification. Traditionally, for a supervised machine learning algorithm such as SVM, an extraction stage is required where features are hand-crafted by experts based on their knowledge in the specific application domain. Attempts have been made in various research fields to find effective features, for example in image-processing-related applications [75]. However, this approach is not always possible for classifiers and it is often practically difficult, for instance when the relationship between input measurements and user-defined classes is extremely complex or even completely unknown beforehand. Additionally, features that are crafted manually maybe not optimal. For this reason, finding more systematic ways to get good features has drawn an increasing research interest [14].

Notable progress has been done recently to find learning techniques that allow models to learn features automatically from data with minimal manual input. Solutions using deep neural networks (NNs) have especially attracted much attention. The effectiveness of deep NNs has been demonstrated in many fields other than image classification, such as audio and natural language processing or transfer learning.

Among deep learning approaches, Long Short-Term Memory recurrent neural networks (LSTM) have been shown to be effective for prediction and classification of time sequences [56]. These particular units of the recurrent network are capable of retaining information for both short and long-term and are reasonably the best choice for applications like speech recognition as in [49, 94].

Various architectures of RNNs are tested in [96] for the construction of a visual terrain classification model outperforming standard visual approaches in dealing with issues such as illumination changes or motion blur. A road surface classification model based on LSTM was also proposed in [99] where 14 sensors output sequences are used together to recognize flat road, sinusoidal road, potholes, and bumps showing optimal results. Vehicle-terrain interaction sound was used in [123] to train and test a deep spatiotemporal terrain classification model. The rover used in [123] is therefore equipped with a shotgun microphone, and LSTM is integrated with a convolutional neural network to recognize nine different types of terrain taking also into account adverse acoustic conditions. Spectrograms of acoustic signals are computed, and a sequence of features is derived from them to be further classified by an LSTM. Results of [123] highlight the importance of temporal dynamics for the terrain-classification task.

Research in [46] compared the performance of several machine learning algorithms, including Support Vector Machine (SVM), Multilayer Perceptron (MLP), Deep Neural Networks (DNN), and Convolutional Neural Network (CNN), for both terrain estimation and slip detection. This research investigates both vision-based and proprioceptive-based classification with filtered and unfiltered data. The image dataset was provided by NASA's Planetary Data System and the University of Almería's Fitorobot, while the proprioceptive data were collected from an MIT single-wheel test-bed equipped with a torque sensor, an IMU, and a displacement sensor.

The absolute value of the wheel torque together with the variance of pitch, longitudinal, and vertical accelerations were used to compose the four-element feature vector defined by the authors' expert knowledge and train all models for comparison. Refer-

ence [46] highlights the advantages of deep learning algorithms for being able to reach good performance with raw data without necessarily requiring expert feature extraction. Researchers in [92] trained a feed-forward Neural Network (NN) for each one of 15 different sensors (3-axes gyros and accelerometers, two motor current sensors and two voltage sensors, one ultrasonic and one infrared range sensors, one microphone, and one wheel encoder). The authors compared the performance of NNs for different sensor modalities for DFT-based classification of five different terrain types (gravel, grass, sand, pavement, and dirt). Results of [92] show that certain sensors are better suited for identifying certain types of terrain and suggest that better performance can be achieved by combining multiple sensor modalities. A recent body of research has been devoted to the classification of terrains in the context of legged robots [57]. In [8], torque measurements taken from sensors attached on robot legs were passed through RNN and LSTM to capture temporal data. Classification of terrain class (high-friction, low-friction, deformable, granular) was discussed in [65] for the SAIL-R legged robot using an SVM with 39 hand-designed features extracted from the ground reaction forces and motor speed. Finally, an unsupervised learning based on the Pitman-Yor process was also presented in [29].

Section 2.3 presents a Convolutional Neural Networks for terrain classification purposes over data collected with SherpaTT rover. The adoption of recurrent and CNN is discussed in 2.4, in the context of terrain classification using the agricultural robot Husky equipped only with inertial and electrical current sensors. Section 2.5 presents a fair comparison between hand-crafted and learned features through a rigid feature scoring and selection process. Addressing the challenges in evaluating the effectiveness of learned features contrasted with expert-designed ones and dealing with the complexity of determining the descriptive power of hand-crafted features.

2.2 Support Vector Machine

SVM is a well-established machine learning solution for soil classification problems [12, 45, 111]. This section will present a summary of the theory behind SVM classification. For a detailed description of SVM algorithm please refer to [55] and [124]. SVM classifies the data attempting to separate them with hyperplanes during training and retains for testing only few samples, called support vectors. The input of an SVM model is a feature vector where the sensory data are generally composed or elaborated with expert knowledge. An SVM problem is composed of two stages: training and testing. Given two classes A and B (binary classifier), an input training set S composed of p samples and n features can be defined as:

$$i = [1, 2, \dots, p] \text{ where } \begin{cases} y_i = 1 & \text{if } \mathbf{x}_i \in A \\ y_i = -1 & \text{if } \mathbf{x}_i \in B \end{cases} \quad (2.2.0.1)$$

\mathbf{x}_i are referred as predictors, y_i represents the response variable. The purpose of the linear SVM algorithm is to find a decision function D that allows, in the testing phase, to classify any new sample $x \in \mathbf{R}^n$ according to the sign of $D(x)$. This is done by finding the hyperplane that maximizes its distance to the support vectors (i.e., the predictors closest to the hyperplane), while minimizing the loss due to misclassification. The Lagrangian dual of this optimization problem can be formulated as follows:

$$\begin{aligned} \max_{\alpha} & \left(\sum_{i=1}^p \alpha_i - \sum_{i=1}^p \sum_{j=1}^p \alpha_i \alpha_j y_i y_j \mathbf{x}_j^T \mathbf{x}_j \right) \\ \text{subject to} & \sum_{i=1}^l y_i \alpha_i = 0 \text{ and } 0 \leq \alpha_i \leq C, \end{aligned} \quad (2.2.0.2)$$

where α_i are Lagrangian multipliers and nCl is a parameter called box constraint. The dominant approach for multiclass applications is to reduce the single problem into multiple binary classification problems [33]. One of the most common methods for such reduction is the error-correcting output codes (ECOC) model [31]. The most important parameter for this method is the coding design, a matrix where elements indicate which classes are trained by each binary learner, reducing the multiclass problem to a series of binary problems.

2.3 Convolutional Neural Networks

In contrast to SVM that uses handcrafted features manually engineered by data analysts, CNN derives features automatically from inputs throughout a training process, searching for those that better characterize each terrain. However, as input, CNN takes an imagelike observation, therefore a first practical issue to solve is how to derive a 3D object from several signals. The proposed solution is to resort to fast Fourier transform (FFT) to construct magnitude spectrograms of the signals then appended into a multichannel object forming the input for the net. So, sensory data can be assembled in 3D shape, namely height, width and depth. The height corresponds to the frequencies (nF) analyzed by the FFT, the width corresponds to the number of time windows (nW) adopted in the spectrogram, and the depth is the number of signals (nCh).

2.3.1 Multichannel Spectrograms

In this thesis, nCh refers to the number of channels that correspond to the available sensors. Considering that any signal in time $s(t)$ for $t \in [t_0, t_1]$ N -tuple $x \in \mathbf{R}^2$ with $N = \lfloor (t - t_0) \rfloor \cdot sf$, with $\lfloor \cdot \rfloor$ indicating the extraction of the integer part of a number. Equation 2.3.1.1 computes the discrete Fourier transform $\hat{x} \in \mathbb{C}^N$ and equation 2.3.1.2 defines the matrix operator \bar{F} .

$$\hat{x} = \frac{1}{N} \bar{F} x \quad (2.3.1.1)$$

$$\bar{F} = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & e^{j2\pi\frac{1}{N}} & e^{j2\pi\frac{2}{N}} & \dots & e^{j2\pi\frac{N-1}{N}} \\ 1 & e^{j2\pi\frac{2}{N}} & e^{j2\pi\frac{4}{N}} & \dots & e^{j2\pi\frac{2(N-1)}{N}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & e^{j2\pi\frac{N-1}{N}} & e^{j2\pi\frac{2(N-1)}{N}} & \dots & e^{j2\pi\frac{(N-1)^2}{N}} \end{bmatrix} \quad (2.3.1.2)$$

The discrete Fourier transform \hat{x} contains the complex coefficients associated with the double-sided spectrum of N frequencies in the range $[0, sf]$. Each sample is represented by a group of nCh signals, $s_c(t)$ for $t \in [t_0, t_0 + MW]$ and $c = 1, 2, \dots, nCh$. Each signal $s_c(t)$ is windowed with windows of length wL seconds and overlapping wO seconds resulting in signals $s_c^w(t)$ and $w = 1, 2, \dots, Nwind$. Equations 2.3.1.3 define the ranges of time $[t_1^w, t_1^w]$ while equation 2.3.1.4 describes computation for the number of windows $Nwind$.

$$\begin{cases} t_0^w = t_0 + (w-1)(wL - wO) \\ t_1^w = t_0^w + wL \end{cases} \quad (2.3.1.3)$$

$$Nwind = \lfloor \frac{\lfloor MW \cdot sf \rfloor - \lfloor wL \cdot sf \rfloor}{\lfloor (wL - wO) \cdot sf \rfloor} \rfloor \quad (2.3.1.4)$$

Considering $x_c^w \in \mathbf{R}^{\lfloor wL \cdot sf \rfloor}$ the vector associated with signals $s_c^w(t)$, equation 2.3.1.1 allows for the computation of the discrete Fourier transform $\hat{x}_c^w \in \mathbb{C}^{\lfloor wL \cdot sf \rfloor}$. Each sample is here transformed in $Nwind$ Fourier transform for every available channel. Given symmetry with respect to the Nyquist frequency $\frac{sf}{2}$ of the double-sided magnitude spectrum $|\hat{x}_c^w| \in \mathbf{R}^{\lfloor wL \cdot sf \rfloor}$, equations 2.3.1.5 define the single-sided magnitude spectrum $|M_c^w| \in \mathbf{R}^{nF}$ with frequencies in the range $[0, \frac{sf}{2}]$ and equation 2.3.1.6 computes the number of frequencies nF .

$$\begin{aligned} & \text{if } \lfloor wL \cdot sf \rfloor \text{ is even} \begin{cases} (M_c^w)_f = |\hat{x}_c^w|_f \quad \forall f = \{1, nF\} \\ (M_c^w)_f = 2|\hat{x}_c^w|_f \quad \forall f = \{2, 3, \dots, nF - 1\} \end{cases} \\ & \text{if } \lfloor wL \cdot sf \rfloor \text{ is odd} \begin{cases} (M_c^w)_f = |\hat{x}_c^w|_f \quad \forall f = \{1\} \\ (M_c^w)_f = 2|\hat{x}_c^w|_f \quad \forall f = \{2, 3, \dots, nF\} \end{cases} \end{aligned} \quad (2.3.1.5)$$

$$nF = \lceil \frac{wL \cdot sf + 1}{2} \rceil \quad (2.3.1.6)$$

where the notation $\lfloor \cdot \rfloor$ indicates the nearest greatest integer of a number. Equation 2.3.1.7 finally defines the multichannel spectrogram $S \in \mathbf{R}^{Nwind \times nF \times nCh}$.

$$S_{w,s,c} = (M_c^w)_f \quad \forall f = [1, \dots, nF] \quad (2.3.1.7)$$

2.3.2 CNN Architecture

The architecture of the CNN is shown in Figure 2.3.2.1, where the neural dimensions and the learnable variables of each layer are indicated. The first layer takes as input the multichannel spectrogram, next, the batch-normalization layer normalizes inside the mini-batch the value kept by each input neuron. Normalization process follows Equation 2.3.2.1 where x_n and y_n are respectively, the input and output values of neuron n of this layer, batch mean μ and SD σ are computed during training, while learnable parameters offset γ and bias β are searched through optimization across the whole training set. Computational constant ϵ can improve numerical stability when variance σ_B^2 is small.

$$y_n = \gamma \frac{x_n - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} + \beta \quad (2.3.2.1)$$

$$\forall n = 1, \dots, (nW \cdot nF \cdot Ch)$$

The following 2D convolution layer spans the output across time and frequency domain convoluting the $nW \times nF \times nCh$ batch-normalized spectrograms into $nFilt$ objects of dimensions $nW \times nF$. A user-specified number of square filters $nFilt$ with size fsz are here used to perform convolution process briefly described in Equation 2.3.2.2, where X is the zeropadded neural grid after batch-normalization and Y the output of the convolution process. Each filter combines all channels into a single object trying to find through the training process the combination of channels that better represents the terrain. In this context, the filter size fsz controls both the frequency and time span in which magnitudes across the channels are combined to form values representative for the terrain. $nFilt$ sets, instead, the number of filters that allow the terrain signature to be adequately mapped between control input (e.g., electric motor currents) and sensory output (e.g., the wheel speeds and IMUs measurements). The learnable parameters of this layer are the kernel of filter m , the weights of matrix ${}_m\omega$, and ${}_m\beta$ the corresponding bias.

$$Y_{w,f,m} = \sum_{c=1}^{nCh} \sum_{i,j=-\frac{fsz}{2}}^{\frac{fsz}{2}} {}_m\omega_{i,j,c} X_{w+i,f+j,c} + {}_m\beta \quad (2.3.2.2)$$

$$\forall w = 1, \dots, nW, \quad \forall f = 1, \dots, nF, \quad \forall m = 1, \dots, nFilt$$

The output of the convolution process is passed to the REctified Linear Unit (ReLU) activated neurons in a grid $nW \times nF \times nFilt$, fully connected to nCl neurons where nCl is the number of terrain classes considered. Compared to other activation functions such as the sigmoidal function, ReLU helps in preventing the exponential growth in the neural network computation and the ‘‘vanishing gradient’’ problem that is the tendency for the gradient of a neuron to approach zero for high values of the input [67]. The two following layers SoftMax and Classification are standard as output layers for classification

networks. The function SoftMax is defined in Equation (5) where x_n is the n th input neuron and y_n is the corresponding output of this layer.

$$y_n = \frac{e^{x_n}}{\sum_{i=1}^{nCl} e^{x_i}} \quad \forall n = 1, \dots, nCl \quad (2.3.2.3)$$

The output layer of this network is the classification layer that computes the cross-entropy loss for classification among terrains.

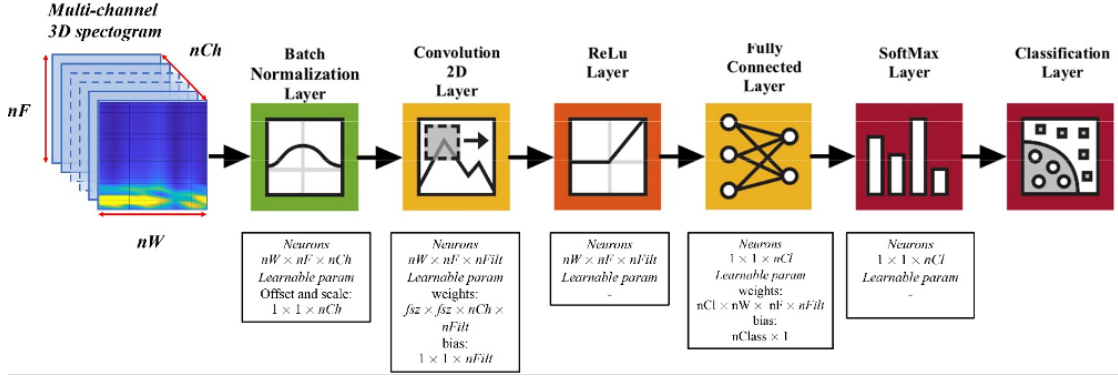


Figure 2.3.2.1: Architecture of the convolutional neural network.

The discussed network structure has been designed to be simple and fast performing, following a classic image classification structure. The proposed algorithm can integrate an arbitrary large set of signals to fuse different sensory output, promoting adaptability to eventual enlargement of onboard sensors. A padding procedure is also proposed to integrate signals provided at different frequencies. Convolution process is performed across time and frequency domain to better capture time-dependent aspects of the rover-terrain interaction. The results obtained concatenating spectrograms in the discussed multichannel object suggest good capability to capture differences and characterizing aspects of different terrains.

2.3.3 SherpaTT rover application

This section of the thesis is based on the published paper [125]. This research is developed within the ADE (Autonomous Decision making in Very Long Traverses) project funded by the European Union to develop novel technologies for future space robotics missions. ADE's objective is to increase the range of traveled distance of a planetary exploration rover up to 1 km/sol, while ensuring at the same time optimal scientific data return. In this context, the ability to sense and classify the type of traversed surface plays a critical role. This section presents a terrain classifier that is based on the measurements of motion states and wheel forces and torques to predict characteristics relevant for locomotion using machine and deep learning algorithms. The proposed approach is tested and demonstrated in the field using the SherpaTT rover, that uses an active suspension system to adapt to terrain unevenness.

Terrain classification represents a challenging task due to high variability in surface type and lighting conditions, possible lack of structure, and no prior information. In this context, heuristic or expert systems may perform poorly, whereas learning approaches may provide better performance [109, 39, 34, 59].

The envisaged idea presented in this research is that terrain properties can be obtained by the rover's wheels that serve as tactile sensors. SherpaTT's wheels are outfitted with six axis load cells that provide a direct measurement of the forces and torques applied by the wheels to the ground and vice versa. In addition, estimation of the motion states in terms of accelerations and rate-of-turn can be gathered by an inertial measurement unit attached to the robot's body. All 20 joints of the suspension system provide telemetry including currents, voltages, PWM, temperatures, velocity, and position. Following this rationale, signals that are directly modulated by the terrain interaction can be obtained. These measurements represent a rich source of information that bring significant content in terms of terrain properties. The work here presented applies learning approaches to this data in order to make intelligent autonomous robots adaptive to the site-specific environment [32, 111].

The general learning process includes gathering of data pertaining to the vehicle-terrain interaction followed by a mapping stage of data with the corresponding terrain. This functional relationship can help addressing various issues: (a) difficulty in creating a physics-based terrain model due to the large number of variables involved, (b) the mapping from proprioceptive input to a mechanical terrain property is an extremely complicated function, which does not have a known analytical form or a physical model and one possible way to observe it and learn about it is via training examples, (c) a learning approach promotes adaptability of the vehicle's behavior. In this research, we use a deep neural network to solve the classification problem of different terrain types. Therefore, sensory signals are fed in the form of multichannel spectrograms to a Convolutional Neural Network (CNN), showing better performance when contrasted with standard Support Vector Machine (SVM). The ground properties detection (GPD) system is tested and developed using the rover SherpaTT that is shown in Figure 1.4.0.1.

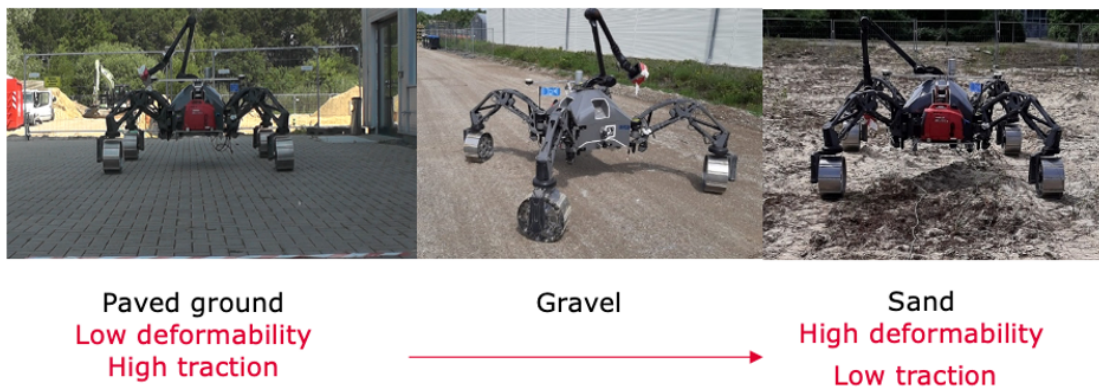


Figure 2.3.3.1: Types of surfaces traversed by SherpaTT during the test and development of the system.

SherpaTT was remotely controlled to follow an approximately 10 m long straight path over three types of terrain: (i) unprepared sandy terrain, (ii) gravel/gravel road, and (iii) paved ground. Sand and paved ground represent the two opposite extremes in a terrain classification scale, since sand can be regarded as a high deformable and low traction surface whereas a paved surface offers low/no deformability and high traction. For each terrain, five runs were repeated in forward drive and five tests in reverse drive. The speed of the rover was controlled as 0.1 m/s and 0.15 m/s. In Figure 2.3.3.1, the three different terrain types are shown taken during the experiments.

All runs were conducted with a fueled power generator mounted to SherpaTT. It can be seen as a red box at the back of the rover in Figure 2.3.3.1. The generator is used for the long traverse in the project as a replacement for solar panels of a flight system, in order to guarantee a 6-8 hour long operational time. Note, that the generator's vibrations might be included in the body's IMU data. Reference runs on battery without vibrations from the generator were also conducted but are not considered yet for the results of this work.

2.3.4 Experimental Results

The deep CNN terrain classifier is validated in the field using real data gathered by SherpaTT operating on different surfaces. The motivation is double-fold.

On one hand, the discriminative power of proprioceptive signals (e.g., inertial and force measurements) is quantitatively evaluated. In this respect, three different classifiers are built: two classifiers that are trained using each singular sensor modality, and one algorithm that combines both sensory data.

On the other hand, the performance of a deep convolutional net is compared with SVM. Raw data collected by SherpaTT with a sampling rate of 100 Hz during straight runs are partitioned in four folders, and then windowed in 1-second adjacent samples. The multimodal observation is then fed in the CNN classifier in the form of multichannel spectrogram, whereas first and second statistical moments are extracted for the SVM embodiment. 80% of the available windowed in each folder is used as training set, whereas the remaining 20% is left out as testing set. Comparison between the two learning approaches is presented in terms of confusion matrices, as shown in Figure 2.3.4.1 and Figure 2.3.4.2.

Figure 2.3.4.1 (a) and (c) shows that an IMU data-based learning approach can give around 85% accuracy. The convolutional neural network, based on signals spectrograms in Figure 2.3.4.1 (a) reaches 88.3% accuracy. Meanwhile SVM, exploiting mean and standard deviation in Figure 2.3.4.1(c) is 4.8% less accurate. Precision of models is also presented in the last column and sensitivity values are contained in the last row. Both models show consistent precision and sensitivity values with lower values on paved ground and gravel, which are terrains with relatively high adherence compared to sand. Lower sensitivity value is relative to SVM capability of discerning paved ground samples from sand ones, based only on IMU signals. Figure 2.3.4.1 (b) and (d) contain results of classification models based only on measurements of forces and torques exchanged between the rear left wheel and the ground. Both SVM and CNN models trained on

signals provided by a single load-cell are quite as much accurate as the corresponding IMU-based ones. CNN performs 0.6% better while SVM is 2.3% less accurate. Variation of accuracy can be explained comparing the four confusion matrices and considering the increasing gravel samples misclassified as sand ones, as shown by both CNN Figure 2.3.4.1 (b) and SVM Figure 2.3.4.1(d). Refer to row Sand and column gravel (red element S-G). This variation is balanced in Figure 2.3.4.1 (b) by CNN model with the increasing correctly classified paved samples (green element PG-PG) previously mistaken in Figure 2.3.4.1 (a) as gravel (G-PG). SVM instead presents in Figure 2.3.4.1 (c) a larger number of misclassified paved samples as gravel (G-PG), thus explaining the reduction in accuracy. Relevant is also the comparison between (b) and (d) of Figure 2.3.4.1.

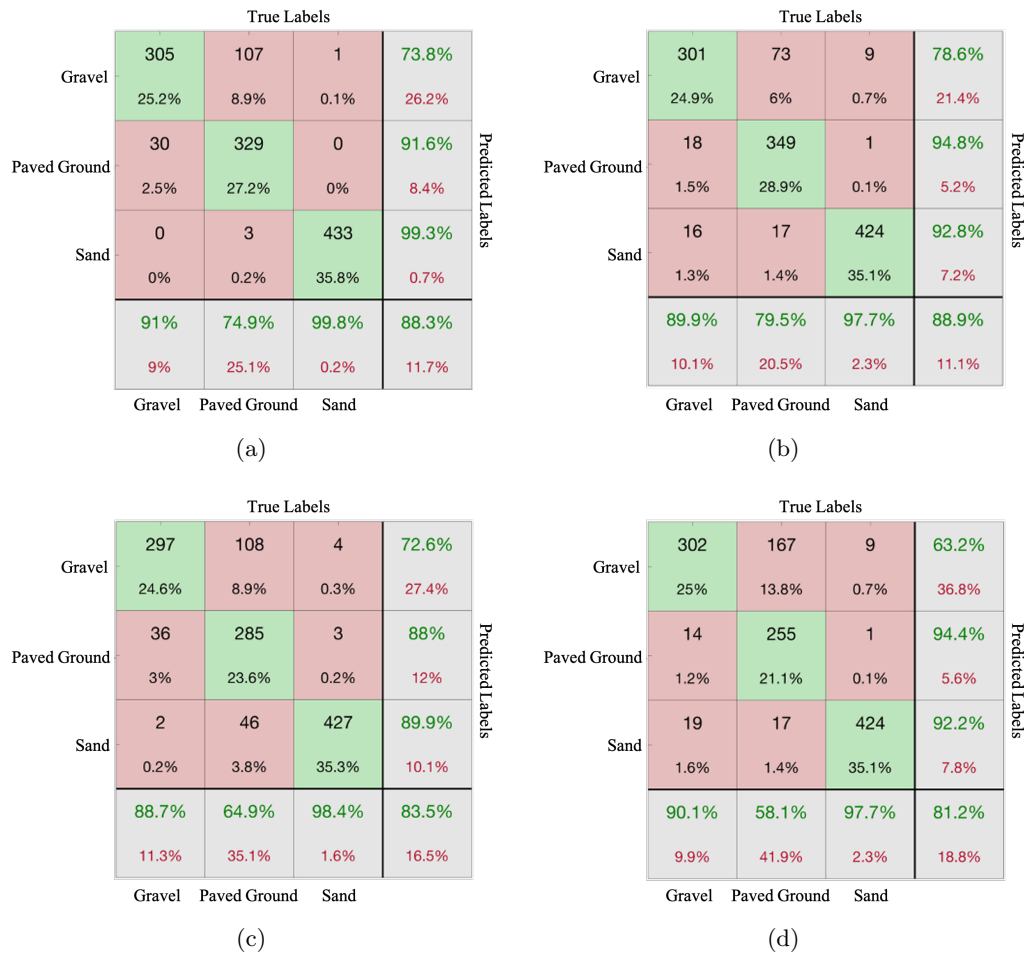


Figure 2.3.4.1: Confusion matrices obtained from (a) CNN trained with IMU data (b) CNN trained with Load Cell data (c) SVM trained with IMU data (d) SVM trained with Load Cell data

The two confusion matrixes not only present about the same number of correctly

classified gravel samples (G-G), but also the same Sand sensitivity: 97.7%. Correspondence between Sand sensitivities of models indicates that the two are equally good at recognizing sand when driving on it. Moreover, correspondence of elements S-PG and PG-S of both (b) and (d) indicates that the two models are equally good at discerning Sand from Paved ground because they mistook the exact same number of paved ground samples for Sand and vice versa. Therefore, providing a wheel with a load-cell will give the rover about the same capacity of discerning between two opposite terrains in terms of traction as Sand and Paved Ground when using a model that relies only on force and torque data.

Figure 2.3.4.2 underlines the importance for a model to be able of fusing the information provided by different sensors. In this respect, the CNN model shows higher capability of fusing IMU and load-cell data to recognize terrain than SVM, reaching 91.4% of accuracy Figure 2.3.4.2(a) outperforming SVM of 6.1% Figure 2.3.4.2(b). Using both IMU and load-cell data still results in higher SVM accuracy than using just one of the two sensors, but SVM does not gain from sensor fusion as much as CNN does. In fact, providing the IMU-based SVM with load-cell data results in a 1.3% more accurate model, whereas providing the load-cell based CNN with the IMU data produces a 2.5% more accurate neural network.

		True Labels									
Gravel		314	80	1	79.5%	Gravel		318	107	0	74.8%
		26%	6.6%	0.1%	20.5%			26.3%	8.9%	0%	25.2%
		21	357	0	94.4%		Paved Ground		16	279	0
	1.7%	29.6%	0%	5.6%		1.3%		23.1%	0%	5.4%	
	0	2	433	99.5%	Sand			1	53	434	88.9%
	0%	0.2%	35.8%	0.5%			0.1%	4.4%	35.9%	11.1%	
	93.7%	81.3%	99.8%	91.4%			94.9%	63.6%	100%	85.3%	
		6.3%	18.7%	0.2%	8.6%			5.1%	36.4%	0%	14.7%
		Gravel	Paved Ground	Sand				Gravel	Paved Ground	Sand	

(a)

		True Labels									
Gravel		314	80	1	79.5%	Gravel		318	107	0	74.8%
		26%	6.6%	0.1%	20.5%			26.3%	8.9%	0%	25.2%
		21	357	0	94.4%		Paved Ground		16	279	0
	1.7%	29.6%	0%	5.6%		1.3%		23.1%	0%	5.4%	
	0	2	433	99.5%	Sand			1	53	434	88.9%
	0%	0.2%	35.8%	0.5%			0.1%	4.4%	35.9%	11.1%	
	93.7%	81.3%	99.8%	91.4%			94.9%	63.6%	100%	85.3%	
		6.3%	18.7%	0.2%	8.6%			5.1%	36.4%	0%	14.7%
		Gravel	Paved Ground	Sand				Gravel	Paved Ground	Sand	

(b)

Figure 2.3.4.2: Confusion matrixes obtained from the CNN (a) and SVM (b) terrain classifier trained combining inertial and force signals.

The ADE project is designed and developed having a set of objectives in mind. Its main objective is to address the current challenges that planetary rover exploration has. ADE is a complex system-of-systems, in which each component is designed to fulfill a specific purpose for reaching the project's objectives. Among the main capabilities provided by ADE is to enable long traverses. To this aim, terrain classification results in a critical component. Data provided during straight forward run by SherpaTT's IMU and rear left wheel cell-load are for this purpose gathered and analyzed. A 4-fold cross-

validation process is carried on among runs and two machine learning algorithms (SVM and CNN) are trained over 1 second long recordings corresponding to approximately 0.1 meters. Classification results are presented for the two models when based only on IMU data, only on load-cell data and with both sensory data. Analysis of corresponding confusion matrices show superiority of the deep-learning approach in classifying unfiltered data and fusing sensory information to provide a better estimate of the traversed terrain with respect to standard SVM-based machine learning classification.

2.4 Recurrent Neural Networks

This section of the thesis is based on the published paper [128]. The future challenge for field robots is to increase the level of autonomy towards long distance (1 km) and duration (1h) applications. One of the key technologies is the ability to accurately estimate the properties of the traversed terrain to optimize onboard control strategies and energy efficient path-planning, ensuring safety and avoiding possible immobilization conditions that would lead to mission failure. Two main hypotheses are put forward in this research: the first is that terrain can be effectively detected by relying exclusively on the measurement of quantities that pertain to the robot-ground interaction, i.e., proprioceptive signals, and no visual or depth information is required. The second is that artificial deep neural networks can provide an accurate and robust solution to the classification problem of different terrain types. Under these hypotheses, sensory signals are classified as time series directly by a Recurrent Neural Network or by a Convolutional Neural Network in the form of higher-level features or spectrograms resulting from additional processing. Results obtained from real experiments show comparable or better performance when contrasted with standard Support Vector Machine with the additional advantage of not requiring an a priori definition of the feature space.

The main contributions of this research to the terrain classification problem for off-road autonomous wheeled robots refer to:

- Reliance on proprioceptive signals that we assume to bring distinctive traits of the terrain as they directly generate from the physical vehicle-ground interaction. Therefore, training a classifier on these signals allows one to identify directly the impact of different terrain types on the vehicle mobility. While proprioceptive sensing has been already proposed generally using one single sensor, here different sensor modalities, e.g., wheel velocities and torques, and inertial measurements, are combined into a single model to achieve robust classification even for terrains with similar properties.
- Use of deep learning to solve the terrain classification problem. While standard machine learning algorithms have been already demonstrated, this research discusses in detail the adoption of deep learning-based solutions. Three different embodiments of deep learning classifiers are presented, namely, Convolutional Neural Network (CNN), Long Short-Term Memory recurrent neural network (LSTM), and Long Short-Term Convolutional recurrent neural network (C-LSTM). While the

two RNNs accept as input directly the time series of the signals, CNN operates through signal spectrogram. Therefore, a direct comparison between two different approaches that use either raw signal or their spectral content is performed.

- A CNN-based terrain classifier is presented using a novel approach where heterogeneous input data are assembled in a multi-dimensional spectrogram. Previous attempts in this direction have generally dealt with a single-channel spectrogram, as for example in [123] or [113]. In contrast, the proposed approach can be extended to any sensor combination without limitation on the number of measurements and their sampling frequency. By combining different sensor modalities, classification performance is significantly improved.
- Parametric analysis of the proposed deep learning classifiers to evaluate the influence of design parameters, including the temporal "listening" window.
- Direct comparison between the proposed deep terrain classifiers with the existing benchmark, e.g., SVM. Such a comparison is seldom discussed in the literature. We expect that the self-learned features found via deep learning may capture the temporal patterns from the data, which are not expressed by the hand-designed feature space adopted by SVM.

2.4.1 RNN Architecture

Considering nCh as the number of channels and Nh as the number of hidden units, $X_t \in \mathbf{R}^{nCh}$ and $Y_t \in \mathbf{R}^{Nh}$ are respectively the input and output vectors at time instant t of a Recurrent Neural Network (RNN). The architecture of an RNN illustrated in Figure 2.4.1.1 consists of a sequence of recurrent units (RUs). Each RU has the same internal structure and outputs a hidden layer vector of size $Nhx1$ represented by h_t together with the output vector. The hidden layer vector h_t is then given as input to the following RU together with the input vector X_{t+1} .

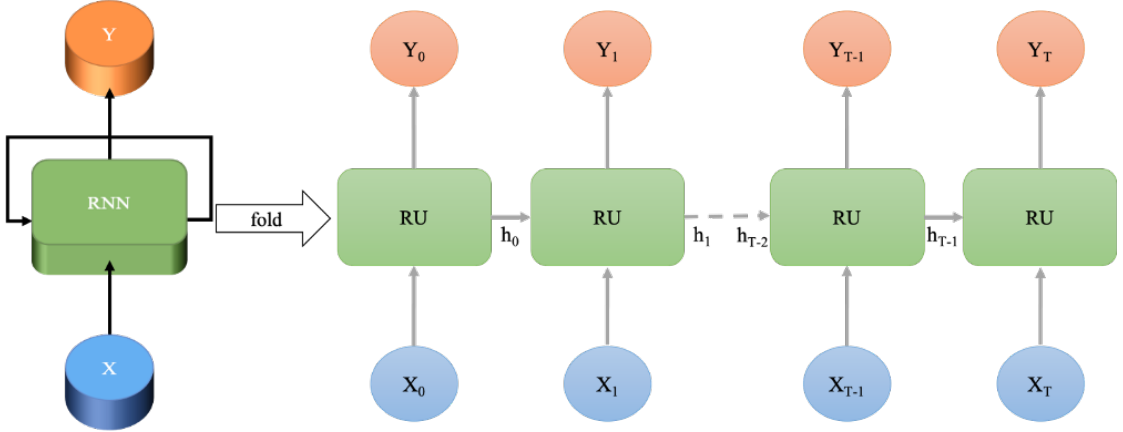


Figure 2.4.1.1: Recurrent Neural Network structure. In this study $X_t \in \mathbf{R}^{10}$, $Y_t \in \mathbf{R}^{15}$.

With sf being the sampling frequency, X is a sequence of $\lfloor T \cdot sf \rfloor$ input vectors meanwhile Y is the corresponding sequence of output vectors. An RNN has two different operating modes, sequence to sequence and sequence to label. The sequence to sequence mode associates the sequence of input vectors to a sequence of output vectors, whereas the sequence to label mode returns the last output vector of the sequence as representative of the entire input sequence. The last output of the sequence Y , namely Y_T is in fact influenced by all the previous inputs and therefore the most representative for classifying the entire sequence. Given the number of classes nCl , the vector Y_T is passed to a fullyconnected layer with a weight matrix of dimension $(nCl \times Nh)$ and bias vector of size $(nCl \times 1)$. The fully-connected layer is further connected with a soft-max layer and a classification layer after that, constituting the classic classification structure. The internal structure of the repeating unit (RU) of a simple RNN is showed in Figure 2.4.1.1 on the left representing the mathematical relationship between the vectors X_T, Y_T and h_T , given:

- W_h and W_y weight matrixes of size $(Nh \times Nh)$
- W_x the weights matrix of size $(Nh \times nCh)$
- b_h and b_y bias vectors of size $(Nh \times 1)$

The values retained by these weights and biases are computed during the training process searching for the optimum point of the cost function. The activation function \tanh is the hyperbolic tangent and constitutes part of the classical recurrent unit of a simple RNN capable of solving simple sequence classification problems. These types of recurrent networks suffer from what is known in literature as problems of vanishing or exploding gradient due to the fact they are not capable of propagating information through time except for the previous time instant.

2.4.2 Long-Short Term Memory RNNs

Long Short-Term Memory RNNs (LSTM) solve vanishing gradients problem by changing the structure of the recurrent unit, using the much more complex structure shown in Figure 2.4.1.1 on the right. It has been proven [56, 49] that LSTM networks are capable of blocking, forgetting and retaining information through the block gate C_t , the forget gate f_t , the input gate i_t and the memory state C_t , all vectors of dimensions $(Nh \times 1)$. The hidden layer vector together with output and input vectors have the same meaning and size as explained before and the size of weights and biases can be derived to make coherent the following operations:

- \otimes row-by-column matrix product
- \oplus sum between vectors
- \odot Hadamard elementwise product between vectors.

The activation function σ our study and depicted in 2.4.2.1 has a classic Sequence to Single class value recurrent structure. The sequence input layer takes a sequence of feature vectors and passes the data to the LSTM layer whose output is the last vector of the hidden units. The LSTM layer output is then passed to a fully-connected layer with dimension nCl . The last combination of the two layers soft-max and classification, output the predicted single-value class associated with the sequence of feature vectors. Different number of hidden units, Nh , have been tested, and eventually increasing Nh beyond 15 did not result in any significant improvement.

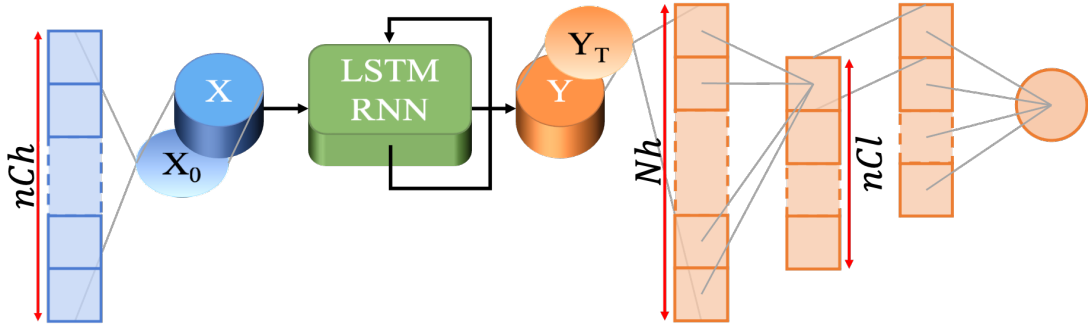


Figure 2.4.2.1: Long Short-Term Memory Model Structure. The number of channels is nCh , the number of hidden units is Nh and the number of terrain classes is nCl .

2.4.3 Convolutional LSTM

The Convolutional LSTM network (C-LSTM) takes advantage of the convolution process among features for terrain classification. The structure of this model showed in Figure 2.4.3.1 is like the previously explained except for the first part. The sequence input layer is in fact followed by a sequence folding layer that considers every feature vector from the sequence and passes it to a two-dimensional convolutional layer that performs the convolution process, similarly to the procedure described for the CNN model. The convolution results are normalized by the batch normalization layer and handed over to the rectified linear unit (ReLU) layer. The sequence unfolding layer then collects the results of these operations performed on every element of the feature vector sequence and together with the flatten layer they pass this new sequence to the LSTM layer. Specifically, the flatten layer reduces the extra dimensionality produced by the convolutional layer.

The following layers of the net structure are identical to the LSTM model structure. This convolution process allows the C-LSTM model to autonomously search relationships between features thus sensor outputs values for classification purposes. Five different filters have been trained and the filter size fsz is set equal to the channels vector length ($nCh = 10$) in order to let the net autonomously select a specific subset of inputs to be related and associated in a new channel.

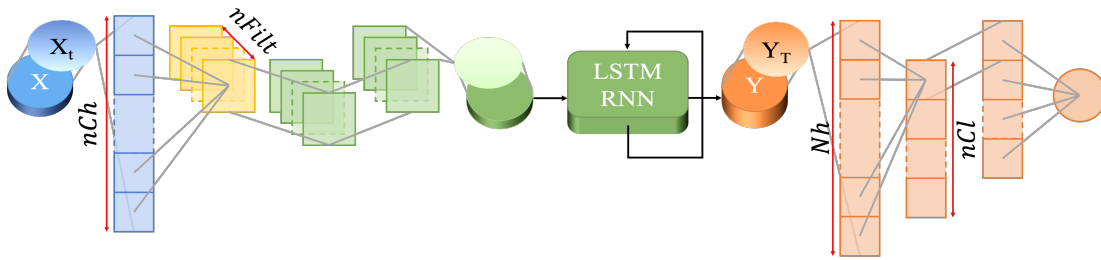


Figure 2.4.3.1: Convolved Long Short-Term Memory Model Structure. The number of channels is nCh , the number of filters is $nFilt$, the number of hidden units is Nh and the number of terrain classes is nCl .

2.4.4 Husky rover application





The proposed terrain classifiers are tested in the field using the experimental testbed Husky that is shown in Figure 1.4.0.4. Its sensor suite exteroceptive counterpart has not been logged for this specific research. Therefore, two groups of measurements can be logged by the robot during operations: the IMU data sampled at 50 Hz and the wheel service data that we refer to as the PRO data sampled at 15 Hz. The IMU data contain the measurements of the 3-axes gyroscope and accelerometer inertial unit. The PRO data instead consist of the electric driving currents of the left and right motors and the left and right wheel velocities derived from the incremental encoder readings.

Sensory data are gathered as Husky traverses four different types of terrain: concrete, dirt road, unploughed, and ploughed, shown in Table 2.4.4.1. Extraction of sample S_i from collected sensor's signals is performed after a partitioning procedure of the data to ensure generality of the models and avoid overfitting, according to the pipeline explained in Figure 2.4.4.1. A partition window (PW) of 5 s is selected and consecutive clips of length PW are extracted from each signal and used randomly for testing (green in Figure 2.4.4.1) and training (orange in Figure 2.4.4.1) sets following a k-fold cross-validation. To augment the number of available data, clips of length PW are then further windowed using a moving window of MW seconds and a stride of ST seconds.

In section 2.4.5.1 different values of MW are tested and the impact on model's performance is investigated. The stride ST is instead adjusted for each terrain type to have a comparable number of samples for each terrain and avoid biasing model's prediction towards those terrains that would present more samples than others in the dataset. Table 2.4.4.1 contains in the third column the total number of windowed 5s long recordings available for each terrain type.

The generality of the presented model performance is achieved through k-fold ($k = 5$) cross validation. Each fold contains 80% of the available windowed data in the training set whereas the remaining 20% is used as a testing set. For each partition, the training set is used to train the classifiers that are afterwards tested on the corresponding testing set.

Table 2.4.4.1: Considered Terrain Types and number of available 5s long windows for each one

Terrain Type	Terrain sample image	Number of 5s long windows
Concrete		24
Dirt Road		16
Ploughed		60
Unploughed		56

For each fold, performance metrics, including accuracy, specificity, precision and F1-score are computed on the respective testing set. In this study $nCh = 10$ and PRO data and IMU data are sampled, respectively, at 15 Hz and 50 Hz. Therefore, down-sampling is performed for the training of RNNs, whereas padding is used for CNN. Four statistical moments of signals (mean, standard deviation, skewness and kurtosis) are computed and appended to form the input feature vector of SVM, instead.

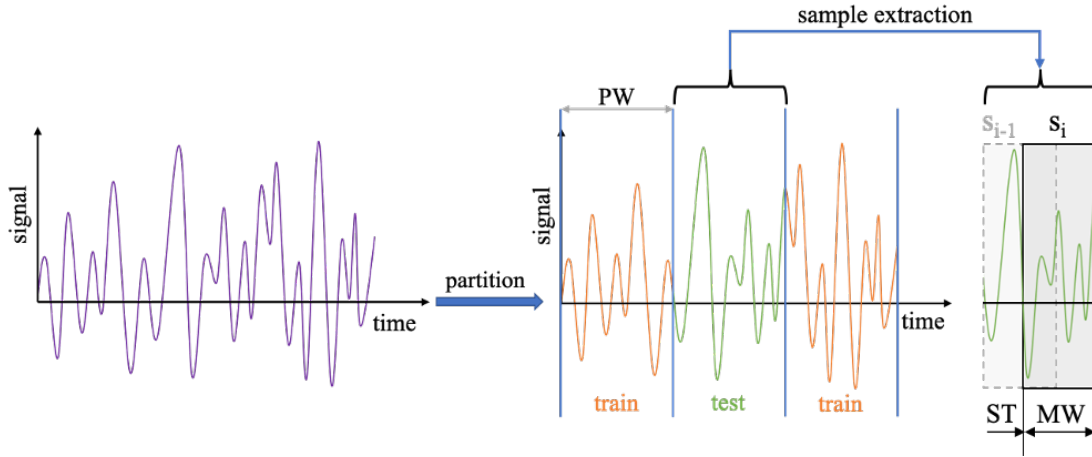


Figure 2.4.4.1: Pipeline of signal partition and sample S_i extraction, with PW the time window used for partitioning, MW the time window used for augmentation and ST the corresponding stride.

Furthermore, equation 2.3.1.6 expresses a direct proportion between the sampling frequency sf and the number of frequencies nF for a certain window length wL . Therefore, magnitude spectra M_c^w corresponding to IMU channels providing data at $sf = 50\text{Hz}$ have more elements than those associated with PRO data sampled at $sf = 15\text{Hz}$. Each element of the magnitude spectra M_c^w corresponding to PRO data is repeated as many times as needed to match dimensions of magnitude spectra derived from channels providing data at faster frequency, in this case IMU. Considering nF as computed with equation 2.3.1.6 for $sf = 50\text{Hz}$, each sample represented by a group of nCh signals of MW seconds is transformed into nCh spectrograms with dimension $Nwind \times nF$, and subsequently rearranged in a single multichannel spectrogram S defined by equation 2.3.1.7.

The number of channels nCh corresponds to the number of magnitude spectrograms contained in a single CNN input sample. Figure 2.4.4.2 shows the magnitude spectra associated with four different channels, two derived from IMU data (first row) and two from PRO data (second row). To underline the different scales of channels, two color-bars are presented on the right-hand side of figure 2.4.4.2, showing the values of acceleration (in ms^{-2}) and current intensity (in Amperes).

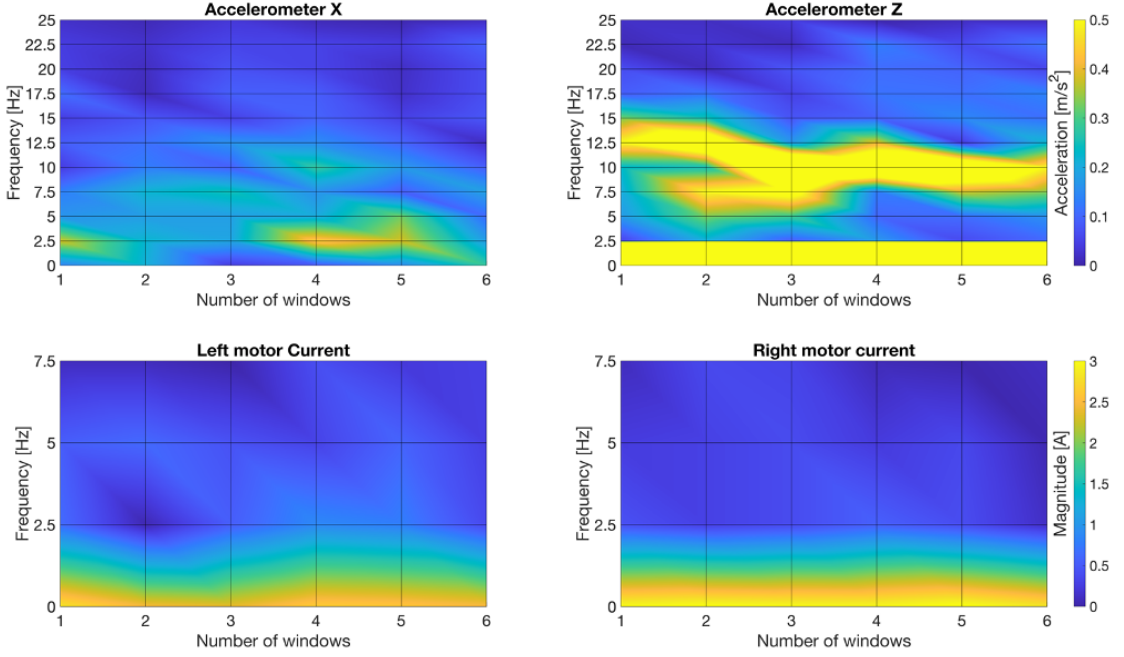


Figure 2.4.4.2: CNN input sample for $nCh = 4$, $MW = 1.5s$, $wL = 0.4s$ and $wO = 0.2s$.

2.4.5 Experimental Results

The results of the three proposed architectures are benchmarked against the established Support Vector Machine (SVM). The SVM model used in this research has a polynomial kernel of fourth-order and "one vs one" coding. It may be further refined and tuned to optimize its performance. However, this is out of the scope of the present research and the interested readers are referred to specific Literature, e.g., [35]. Architecture relevant parameters have been empirically optimized for the 3 Deep Learning algorithms and are reported in Table 2.4.5.1

Table 2.4.5.1: Architecture parameters of DL algorithms.

DL algorithm	parameter	value
CNN	fsz	3
	nFilt	5
LSTM	Nh	15
C-LSTM	Nh	15
	nFilt	5

The performance of the three deep terrain classifiers is quantitatively evaluated using real data acquired in a commercial vineyard. First, the impact of the moving window (MW) size is investigated with a variation range from 0.5 s to 2 s every 0.1 s corresponding to patches from 25 cm up to 1 m. Then, standard classification metrics are

calculated including accuracy, sensitivity, precision, and F1-scores. Normalized confusion matrixes are presented for performance evaluation corresponding to the MW that ensures the best results. Finally, computational burden and memory occupancy of all tested models are discussed to assess the feasibility for online implementation.

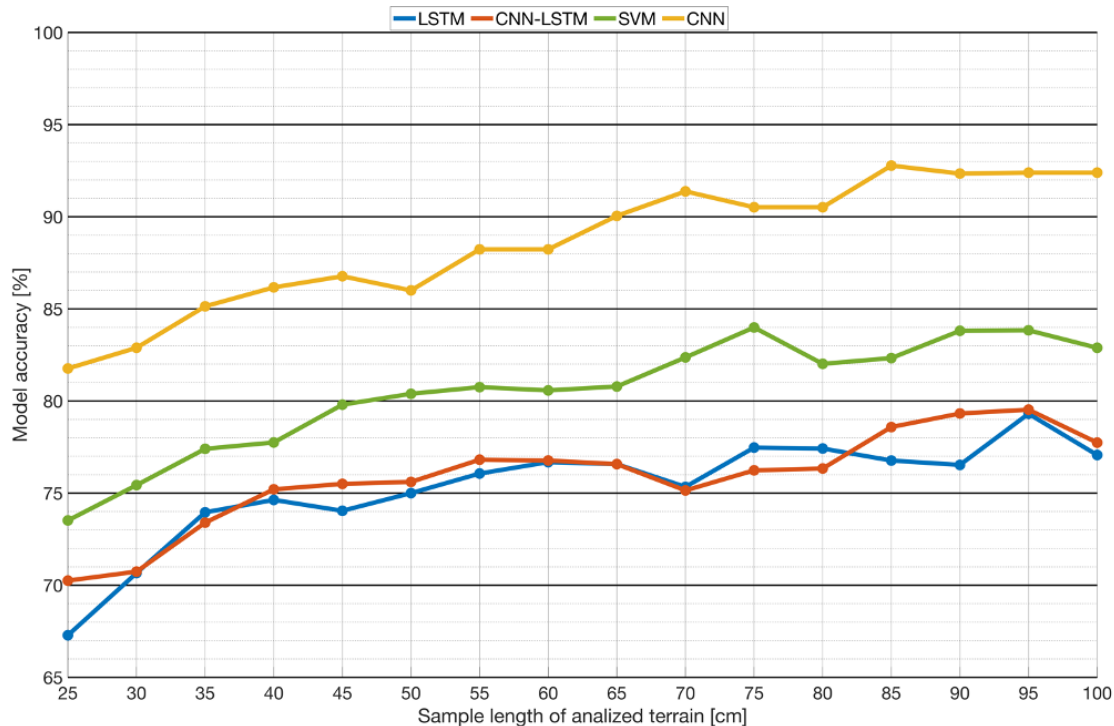


Figure 2.4.5.1: Accuracy of respectively LSTM (blue), C-LSTM (red), SVM (green) and CNN (yellow) models for each sample length of terrain tested corresponding to different SL values. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

2.4.5.1 Parametric analysis

The accuracy among 5-fold partitions is plotted in Figure 2.4.5.1 for each proposed model and moving window between the range 0.5 s and 2 s. The x-axis of Figure 2.4.5.1 contains the sample lengths expressed in centimeter. Each sample length (SL) corresponds to the odometry distance traversed by the robot at the constant speed of 0.5 m/s for a specific moving window (MW) period.

The larger the window the better the accuracy. This can be explained when considering that increasing the MW size results in a larger informative content injected in the classifier (Park et al., 2018). CNN model leads to better accuracy than all other models at every MW . The mean accuracy of the CNN model is already above 80% for a sampling window of 0.5 s that corresponds to approximately only 25 cm of traversed terrain, and it sets to 90% for sampling windows 1.2 s (i.e., 60 cm).

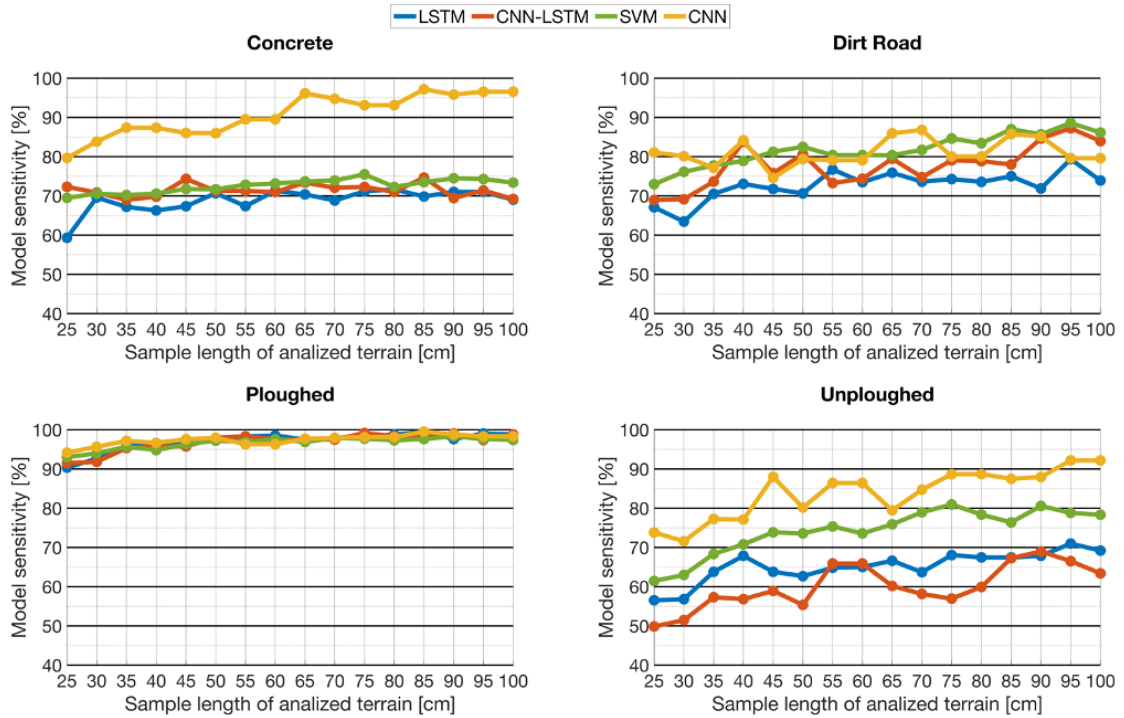


Figure 2.4.5.2: Sensitivity of the LSTM, C-LSTM, CNN and SVM models for each MW tested.

Compared to CNN, SVM model correctly classifies at least 5% less of the available samples, starting with accuracy values lower than 75% for $MW = 0.5$ s and reaching 80% for $MW = 0.9$ s. For SL equal to 85 cm, CNN shows the highest accuracy of 92.8% that is 10.5% better than SVM. LSTM and C-LSTM models present similar accuracy values both always lower than SVM's.

The similarity between LSTM and C-LSTM results suggests that combining instant feature values into a sequence of filtered data does not lead to better performance. For $SL = 85$ cm, the LSTM and C-LSTM models correctly perform 16% and 14.2% worse than the CNN model, respectively. C-LSTM Accuracy values contained in Figure 2.4.5.1 are a good way to understand the influence of MW on each model performance.

Increasing values of MW generally lead to better 5-fold model's accuracy for small values of MW whereas, for $MW = 1.2$ s each model's accuracy fluctuates around a certain value. Mean values and standard deviation for each model accuracy corresponding to $MW \in [1.3s, 2s]$ (or $SL = 65$ cm) are contained in 2.4.5.2. Standard deviations and mean accuracy values in 2.4.5.2 suggest that not only the CNN model is more accurate than the others, but it is also more stable and less subjected to variations of MW for values 1.2 s (or $SL = 65$ cm).

The distribution of sensitivity and precision is presented in 2.4.5.2 and Figure 2.4.5.2. All models are perfectly able to recognize plowed terrain, independently of the MW and present difficulties in singling out concrete terrain and unploughed.

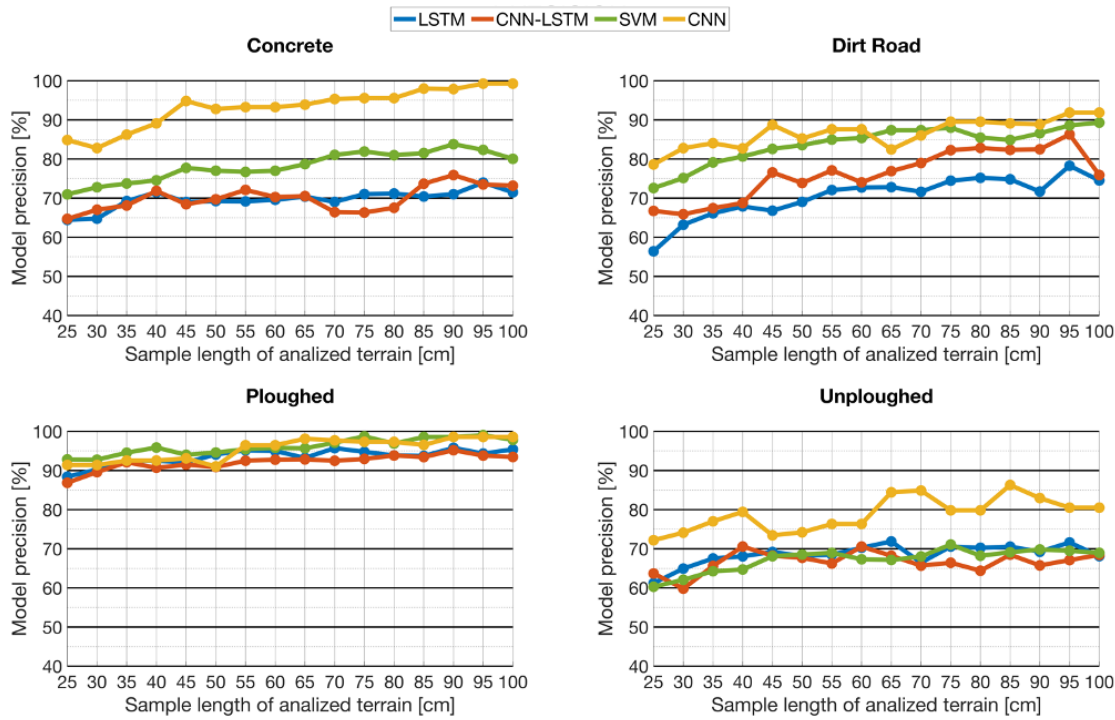


Figure 2.4.5.3: Precision of respectively LSTM, C-LSTM, CNN and SVM models for each *MW* tested.

Precision and sensitivity can be combined in terms of F1-score as shown in 2.4.5.3. Sensitivity and Precision values are well balanced for all the models and all the classes; therefore, class specific performance can be evaluated analyzing F1-score in 2.4.5.3.

Table 2.4.5.2: Mean and standard deviation for accuracy values corresponding to the *MW* range [1.3s, 2s] (i.e. [65cm, 1m]).

	LSTM	C-LSTM	SVM	CNN
Mean Accuracy	77.1%	77.4%	82.8%	91.5%
Accuracy standard deviation	1.13%	1.61%	1.10%	1.08%

In particular, the CNN model outperforms SVM in classifying concrete samples, but show approximately the same F1-score values for dirt road. CNN model also presents better F1-scores for unploughed terrain where the recurrent models present difficulties. Dirt road F1-scores corresponding to recurrent models indicate better performance for the C-LSTM than LSTM. As seen from Figure 2.4.5.1, the best accuracy is obtained by the CNN model for a value of the *MW* size of 1.7s(or SL = 85 cm),which is therefore adopted as the preferred design choice in the classifier implementation.

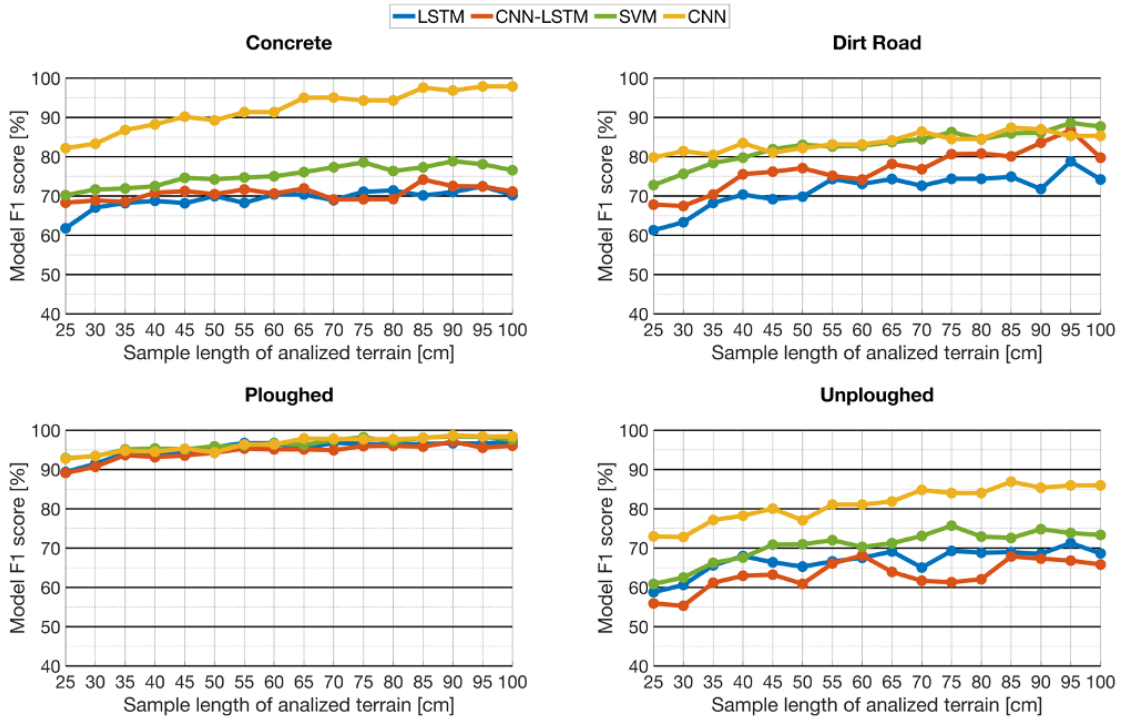


Figure 2.4.5.4: F1-score of respectively LSTM, C-LSTM, CNN and SVM models for each MW tested.

2.4.5.2 Performance evaluation

The normalized confusion matrixes corresponding to the 5folds results are presented for all models in 2.4.5.5 for a value of $MW = 1.7$ s, that led best results and is therefore used for further considerations. Analysis of normalized confusion matrixes outlines what classes are mistaken for each other. Diagonal elements of normalized confusion matrixes contain the sensitivity value of the corresponding class.

As expected, the CNN presents better overall precision and sensitivity values for all classes, showing some difficulties only in sorting out dirt-road samples and unploughed ones. Concrete, dirt road and unploughed terrain are three similar compact terrains, LSTM, C-LSTM and SVM models present difficulties to sort them out from proprioceptive data. Instead, the CNN model proved to be particularly good in recognizing concrete samples, presenting higher values for both precision and sensitivity of this class.

2.4.5.3 Computational burden

The online implementation of a proprioceptive-based terrain classification model in the onboard processing unit of an autonomous robot requires low memory usage and fast performing. The computational burden of all trained models is therefore analyzed. Results are shown in 2.4.5.3 and they were estimated using a CPU intel i-9 working

at 2.4 GHz. The overall classification time for a single observation calculated as an average value over the testing dataset is reported for all models in the third row of the table along with the corresponding average k-fold storage space (fourth row).

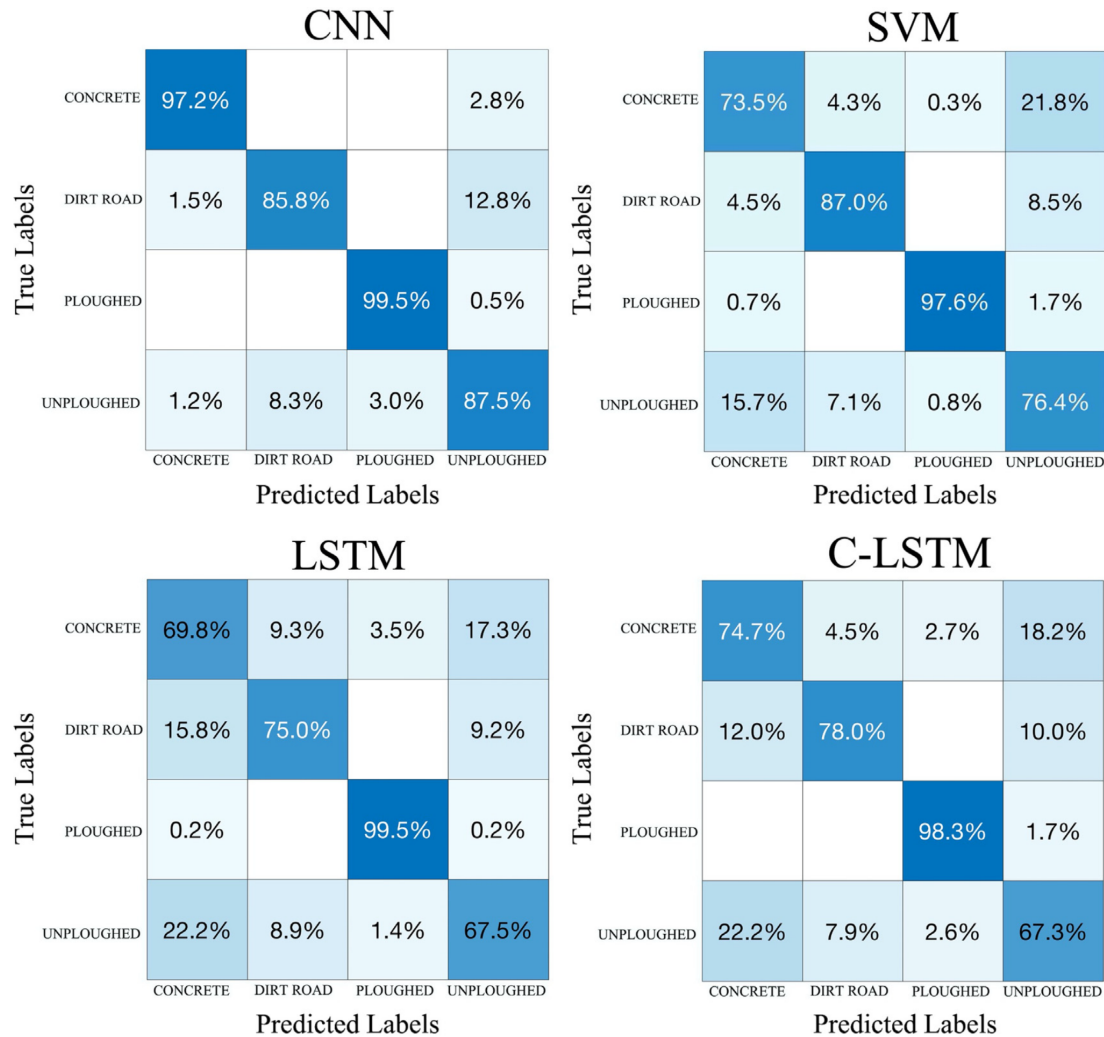


Figure 2.4.5.5: Normalized Confusion Matrixes for all models and $MW = 1.7$ (SL = 85 cm).

The time required for the analysis of a given observation can be divided into a feature generation time followed by a classification time. In this respect, RNNs offer an advantage since they do not require any feature generation stage beside being able to give predictions for every instant of acquired raw data. It should be also noted that SVM entails a preliminary stage for the feature space design that can absorb a significant amount of time and it is subject to a large extent on the user expertise. Although,

this "supervised" preparatory stage is overlooked in this analysis, it represents a clear limitation of SVM when compared to CNN.

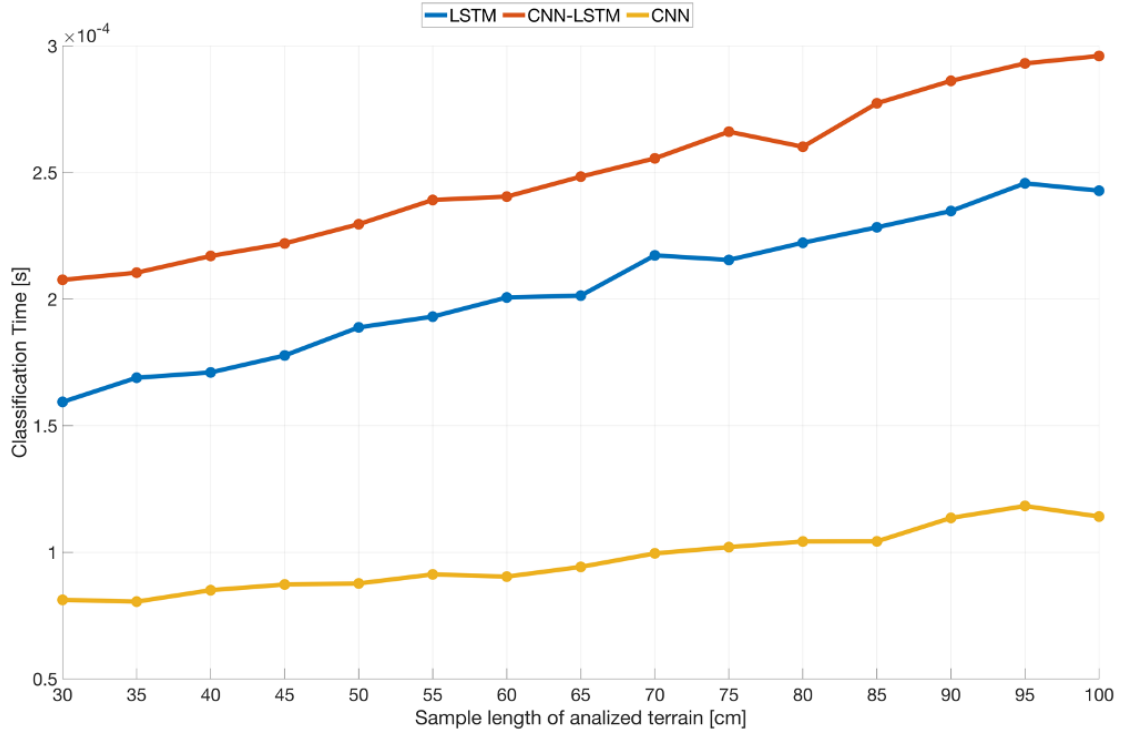


Figure 2.4.5.6: Average classification time as a function of the sample length LSTM LSTM (blue), C-LSTM (red), SVM (green) and CNN (yellow) models for each sample length of terrain tested corresponding to different SL values.

SVM model needs less time to predict the terrain type but weights more on the memory. Memory space occupied by SVM model depends on how much support vectors are retained to classify a dataset and varies with types of features extracted from signals. Highly non-linear problems correspond to a vast amount of support vectors to be retained hence occupying larger space. On the contrary, storage space occupied by RNNs depend only on the engineered structure, simple recurrent structures result therefore in lighter models for resolving highly non-linear classification problems. It is noteworthy also that classification time required by RNNs is highly dependent on sequence length. Longer sequences require more calculations to be classified due to the recurrent structure of the model. Memory space occupied by CNN model depends on the input size of the multichannel spectrogram that is related to sequence length as well.

Figure 2.4.5.6 shows relationship between sample length (SL) and required classification time for neural networks models. From SL = 30 cm to SL = 1 m the samples are quite representative of terrain and increasing their dimension results in more computation required for classification. As it can be seen from the slope of curves in Figure

2.4.5.6 this effect is more significant for RNNs than it is for CNN. Increasing SL from 30 cm to 1 m results in 0.083 ms more required by C-LSTM, 0.088 ms more for LSTM and 0.0329 ms more for CNN. Larger values of MW result in higher computation time required also for feature generation. Build the feature vector based on statistical moments for SVM requires at least 0.095 ms for $MW = 0.5$ s and 0.157 ms for $MW = 2$ s. Assembling the multichannel spectrograms computed with FFT requires the largest amount of time for CNN classification, ranging from 0.665 ms for $MW = 0.5$ s and 2.6 ms for $MW = 2$ s.

Table 2.4.5.3: Model’s computation burden for the classification of a single observation. For all models it is assumed $MW = 1.7$ s (i.e., $SL = 85$ cm).

	SVM	CNN	LSTM	C-LSTM
a: Feature generation [ms]	0.140	2.157	0	0
b: Classification [ms]	0.013	0.104	0.228	0.277
a+b: Total computation time [ms]	0.153	2.261	0.228	0.277
Storage space: Classification [MB]	0.671	0.025	0.020	0.314

2.4.5.4 Overview

This research investigated the use of three deep learning strategies to tackle the terrain estimation problem using proprioceptive data. Ten sensory measurements were used to feed models that predict the traversed terrain after a sufficient listening time. Sensor’s channels efficient combination plays a key role in the terrain estimation task. Best performance was obtained using FFT to compute spectrograms of available signals for every terrain patch and assemble them in a multichannel image within a convolutional network. The CNN model correctly classified up to 92.8% of the observations, 10.5% more than standard SVM concatenating statistical moments, 16% more than LSTM and 14.2% more than C-LSTM. In the existing Literature, higher performance for SVM were reached through extraction of terramechanics parameters often requiring expensive sensors like load-cells for forces and torques on wheels or unavailable ones like GPS for wheel slippage. Nevertheless, SVM proved to be faster than all tested models although RNNs can give real time estimate of traversed terrain with lower accuracy. The largest accuracy was achieved for 1.7 s of sensor’s recordings or approximately 85 cm of traversed terrain. Required computations for prediction took the CNN model 2.26 ms on average making it the slowest of tested models but still suited for online prediction, especially because the use of a dedicated GPU or low-level programming would boost computation times. The proposed proprioceptive-based CNN along with the feature extraction and organization proved advantageous under a variety of aspects. Besides being more accurate than other deep models while being fast enough for online terrain implementation, it can be fed with unfiltered data, hardly classifiable with SVM (Gonzalez and Iagnemma, 2018). Future developments will deal with the study of scalability and portability to other robots of different size, weights, and locomotion systems. Strategies

for self-supervised learning will be also pursued where the vehicle starts its operations with no a priori knowledge of the models that are built progressively during motion.

Software repository The codes and data used for this research are publicly available at this repository [GitHub T_DEEP](#)

2.5 The role of feature and signal selection

This section of the thesis is based on the published paper [120]. Increasing the terrain awareness of planetary exploration rovers is one key technology for future space robotics to successfully accomplish long-distance and long-duration missions. In contrast to most of the existing algorithms that use visual or depth data for terrain classification, the approach presented in this study tackles the problem using proprioceptive sensing, for example, vibration or force measurements. The underlying assumption is that these signals, being directly modulated by the terrain properties, are well descriptive of a given surface. Therefore, terrain signature can be inferred via learning algorithms that are trained on either the signals directly or a signal-derived feature set. Following the latter approach, first, a physics-based signal augmentation process is presented that aims at maximizing the information content. Then, a feature selection algorithm based on a scoring system and an iterative search is developed to decrease the computational cost while preserving high classification accuracy. The resulting most informative feature subspace can be used to train a support vector machine (SVM) classifier. For comparison, the time histories of the selected proprioceptive signals are used to train a deep convolutional neural network (CNN). Results obtained from real experiments using the SherpaTT rover confirm that proprioceptive sensing is effective in predicting terrain type with an accuracy higher than 90% for both algorithms in generalization tasks. When the two learning approaches are contrasted in extrapolation problems, for example, predicting observations acquired at previously unseen velocity or terrain, CNN outperforms the standard SVM. Furthermore, CNN holds the additional advantage of learning features automatically from signal spectrograms, reducing the need of a priori knowledge at the expense of higher computational efforts.

One of the contributions of this study refers to the selection of the most informative subset of proprioceptive features derived from the sensor suite integrated onboard of planetary exploration rovers. A range of aspects is addressed that includes feature extraction, feature ranking, multivariate feature selection, and efficient feature space construction. While feature selection has been largely investigated in other domains, for example, image processing, text processing, and gene expression analysis [53], it remains largely under-investigated for the terrain classification problem of planetary exploration rovers, and rough-terrain robots, in general. In contrast to other areas of applications where datasets with tens or hundreds of thousands of variables are available forming a statistically significant population, data acquired by a rover driving over natural terrain present many challenges such as sparseness, presence of unknown and uncontrolled disturbances, dependence on the specific time and site of the acquisition.

The objectives pursued by feature selection include improvement in the prediction performance, reduction in training time, computational burden and memory usage of the algorithm, and facilitation of understanding the underlying process that generated the data. The other contribution of this study is the adoption of a suitable learning algorithm to infer the type of terrain from the selected feature set. This algorithm will have to look for patterns in the data to construct the mapping from the proprioceptive measurements to the corresponding terrain type.

The well-known support vector machine (SVM) is contrasted with a deep convolutional neural network (CNN). While SVM requires in input hand-crafted features that are selected during a pre-processing stage, CNN uses learned features that are extracted automatically from the signal time histories.

An important goal of the proposed approach is to improve the performance of terrain classifiers for two use cases: generalization and extrapolation. Generalization is defined as the performance of an algorithm on previously unseen observations (test set) that is extracted from the same distribution as the data in the training set, for example, the same test run. The error measured on the test set corresponds to both the online performance of the model and the operating conditions included in the training set. The second use case, extrapolation, is even more challenging since, in general, learning algorithms are known to perform poorly outside the training data population. We compare the performance of the two terrain classifiers (SVM and CNN) for both generalization and extrapolation.

To sum up the novel additions of this study are:

- A whole new signal engineering stage is introduced to improve the overall information content. The signal selection strategy is formalized and reflected in an explanatory block diagram. Improved robustness has been achieved by increasing the number of training repetitions for each candidate feature set;
- The importance of feature selection for terrain classification is shown by comparing a machine learning approach (SVM) with a deep CNN in terms of model complexity, computational burden, and prediction accuracy over a larger terrain set (three types of terrain against two of the previous work);
- The system is evaluated not only in a standard generalization problem but as well as in two more challenging extrapolation contexts that are seldom described in the Literature.

Section 2.5.2 presents SherpaTT, the rover used as testbed, and explains the dataset. Signal augmentation, feature extraction, and selection problems are tackled respectively in Sections 2.5.3, 2.5.4 and 2.5.4.1. The results obtained from the terrain classifiers are presented and discussed in Section 2.5.5. Conclusions wrap up the paper. In this study, SVM is considered as the benchmark approach and compared against the alternative deep CNN.

2.5.1 Learning Parameters

In this section, the values assigned to the parameters of the learning algorithms are highlighted. The parameter set of the SVM-based classifier is indicated in Table 2.5.1.1. It was found empirically to give the best balance of sensitivity and specificity [74]. As for CNN, during the training stage the learnable parameters are updated at each iteration, whereas the hyper-parameters are defined by the user to govern the training process. In one iteration, the network analyses the samples contained in the mini-batch. One epoch consists in the number of iterations necessary to review the entire training data set. The training stage stops after the network has passed through the entire data set the number of times specified as the maximum number of epochs. It is usually preferred to stop the training before this number has been reached, not only because it shortens the time required for training, but also because it prevents overfitting on the training set. Therefore, a percentage of the training data is kept apart as a validation set, and the network evaluates its loss after the number of iterations specified as validation frequency. The validation patience is the number of times that this loss can be smaller or equal to the previously smallest loss before the training stage stops. The initial learning rate drops by a factor (learn drop factor) after a given number of iterations (learn drop period). Part of the hyperparameters is set according to the Literature, for example, the solver and the gradient threshold follow the value suggested in [67].

Table 2.5.1.1: Learning parameters of SVM Classifier.

SVM Classifier Hyper-paramter	Value
C (Box constraint)	1
Standardize	True
Coding design	one-vs-one

Table 2.5.1.2: Learning parameters and Architecture parameters of CNN.

CNN Hyper-paramter	Value
Filter size (<i>fsz</i>)	[5,5]
Number of filters (<i>nFilt</i>)	9
Mini-batch size	160
Maximum number of epochs	150
Validation percentage	15%
Validation frequency	20
Validation patiaence	15
Initial learning rate	0.005
Learn drop factor	0.2
Learn drop period	10

The remaining parameters have been selected empirically through gridsearch and they are reported in Table 2.5.1.2. Note that for a fair comparison with SVM, the

magnitude spectrograms of the signals used as input to CNN are obtained from a time window $w_s = 2$ s (please refer to Section 2.5.4).

2.5.2 SherpaTT rover application

A unique feature of Sherpa is a six-axis force-torque sensor (FTS) mounted on the flange of each wheel-drive actuator, providing direct measurement of the force system exchanged with the ground. The rover also features a six-DOF manipulation arm. The arm is designed to withstand a good portion of the rover’s weight to support it during locomotion. However, for the experiments described in this article, the arm was not involved in locomotion testing. The logging system provides data at a rate of 100 Hz and comprises the following main proprioceptive blocks:

- IMU;
- Wheel-mounted six-axis load cell (LC). In this study, we adopt solely the LC mounted on the front left wheel;
- Joint telemetry (JT). Each of the 20 actuated joints of the suspension system delivers telemetry such as supply voltage, supply current, temperatures, PWM duty cycle, position (relative and absolute), and velocity.

The main data set used for this study was generated at the DFKI premises in Bremen, Germany. SherpaTT was remotely controlled to move for approximately 10 m in a straight line over three types of terrain: sand, gravel, and paved ground. This represents a varied data set with high traction, low deformability surface (paved ground) at one end, and a surface with low traction and high deformability (sand) on the other end, with gravel in the middle of the two (Figure 2.3.3.1). For each terrain, five runs were repeated in forward and reverse drive, except for gravel for which only four runs are available. Two different drive speeds of the rover were used, namely 0.1 and 0.15 m/s.

Table 2.5.2.1: List of available proprioceptive signals.

Signal	Symbol	Sensors	Signal ID
Longitudinal force	F_x	LC	S1
Vertical force	F_z	LC	S2
Drive torque	T_d	LC	S3
Drive electrical current	C_d	JT	S4
Drive PWM duty cycle	PWM_d	JT	S5
Longitudinal acceleration	a_x	IMU	S6
Laterla acceleration	a_y	IMU	S7
Vertical acceleration	a_z	IMU	S8
Gyro roll rate	$gyro_x$	IMU	S9
Gyro pitch rate	$gyro_y$	IMU	S10
Gyro yaw rate	$gyro_z$	IMU	S11

A second data set was generated in a sand mine close to Bremen (please refer again to Figure 1.4.0.1 (b), GPS coordinates (DMS format): 53° 18' 54.9" N, 8° 41' 17.3" E) during the ADE's final testing in April 2021. This independent data set is used to predict terrain labels for observations outside the training data population. In this last environment, the surface traversed was somewhat like the sand case of the previous settings but the terrain was more compact and wetter. It can be directly observed in Figure 1.4.0.1 (b) how humid sand got matted to the wheels while traversing, unlike in the previous environments (Figure 2.3.3.1).

A list of measurements available from the SherpaTT's sensor suite is shown in Table 2.5.2.1, with corresponding sensorial group and Signal ID. From the first analysis of Table 2.5.2.1, some of the signals may appear seemingly correlated. However, if we consider, for example, body acceleration and wheel force, these signals are actually uncorrelated through the flexibility of the suspension system, and therefore they are both relevant for the proposed analysis.

Signals that are directly derived from measurements are referred to as direct signals. Conversely, signals engineered with expert knowledge combining direct signals are referred to as indirect, as explained in the next section. Figure 2.5.2.1 shows a sample time history of the vertical acceleration (gravity-compensated) and drive torque experienced by SherpaTT on different terrains. As seen from this figure, signals show a signature that seems to change according to the specific surface. The goal of this study is to learn this signature to gain terrain awareness. To this aim, it is necessary to select the most relevant signals for building an accurate predictor.

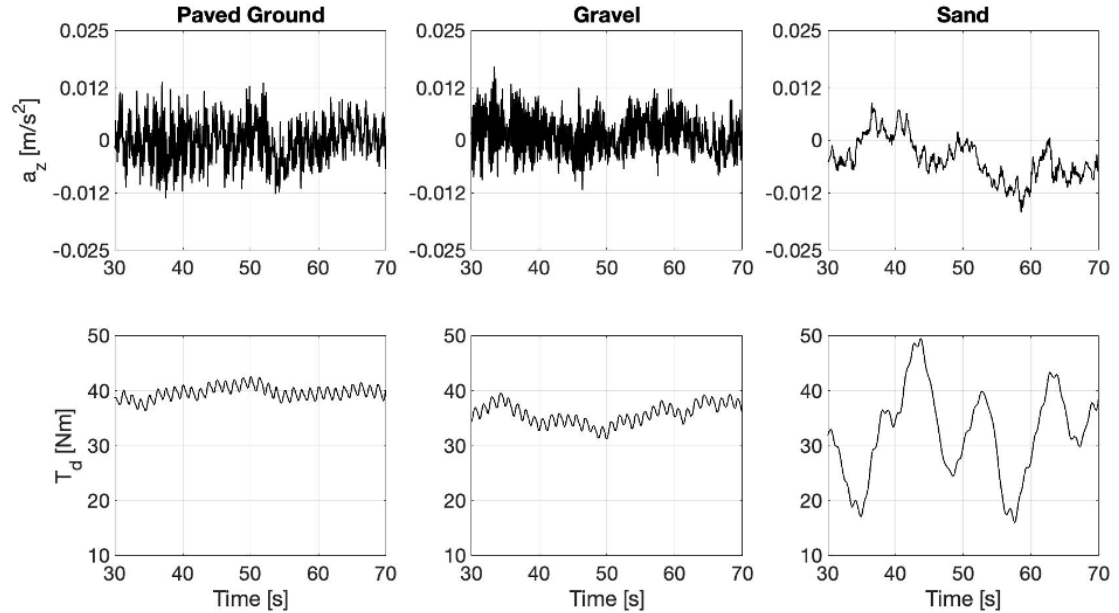


Figure 2.5.2.1: Vertical acceleration and drive torque (wheel front left) measured while SherpaTT driving straight on different terrains.

2.5.3 Signal Augmentation

To improve the information content, an augmentation engine combines multiple direct measurements based on our understanding of the physical mechanisms underlying the wheel-terrain interaction. These are a few of the many possible signal combinations that can be implemented, and they are chosen following a trial-and-error approach to provide the best performance over other alternatives. In this way, nine more indirect signals can be obtained (Table 2.5.3.1). The derivation of these signals is detailed in this section, and the rationale behind the choice of these entities is also explained. Two main motivations support the proposed augmentation stage. First, two or more signals that are useless (not relevant) for themselves can be useful when combined. Then, noise reduction and consequently better class separation may be achieved by adding variables that are seemingly redundant [53]. This explains why we resort to indirect or combined signals and include redundant measurements of the same physical quantity. The first indirect signal is the power loss due to the wheel traction on a given terrain. It can be derived from a “mechanical” or “electrical” analysis. The mechanical power can be estimated as follows:

$$P_M = T_d \cdot \omega \quad (2.5.3.1)$$

where ω is the rotational speed of the wheel. Conversely, the electrical power consumption can be obtained as follows:

$$P_E = \eta \cdot V_d \cdot PWM_d \cdot C_d \quad (2.5.3.2)$$

where V_d is the drive voltage, C_d is the wheel drive current, PWM_d is the duty cycle of the wheel drive Pulse Width Modulation, and η is the efficiency of the electric motor, assumed to be constant and approximately equal to 0.85. Due to the rolling resistance, the direction of the resultant vertical force F_z might not pass through the centre of the wheel, with an offset in the direction of the movement (Figure 2.5.3.1). This is especially true for soft terrain where the impact of rolling resistance is larger. Therefore, we can define the vertical force offset dx from the equilibrium of moments around the centre of the wheel, neglecting the contribution of rotational inertia:

$$dx = \frac{T_d - F_x \cdot R}{F_z} \quad (2.5.3.3)$$

where R is the loaded wheel radius defined as:

$$R = R_N - \frac{F_z}{k_z} \quad (2.5.3.4)$$

being R_N (=200 mm) the nominal wheel radius, and k_z the vertical stiffness of the SherpaTT wheel that was experimentally estimated as 69N/mm. The friction coefficient is an important entity related with the traction ability over the traversed surface. In this study, it is estimated in three different ways:

$$\mu_1 = \frac{F_x}{F_z} = \frac{T_d}{F_z \cdot R} \mu_3 = \frac{C_d k_T}{F_z \cdot R} \quad (2.5.3.5)$$

where k_T (17.4 Nm/A in our case) is the scale factor taking into account the torque constant of the electric motor and the transmission ratio of the motor reducer. Speed deviation is the difference between the angular speed of each wheel ω and the average angular speed of the four wheels $\bar{\omega}$. In this study, speed deviation was estimated in two ways:

$$\begin{aligned} SD &= |\omega - \bar{\omega}| \\ SD_{normalized} &= \frac{\omega - \bar{\omega}}{\omega} \end{aligned} \quad (2.5.3.6)$$

Wheel sinkage is another critical parameter related to rough terrain mobility that can be approximated as suggested in [52]:

$$z = R \cdot \left(1 - \cos \left(2 \cdot \frac{dx}{R} \right) \right) \quad (2.5.3.7)$$

One important aspect is the general data consistency. As an example, Figure 6 shows the drive torque delivered by the leftwheel drive motor, measured by three different sensors. Direct torque measurement from the wheel-mounted LC is denoted by a solid grey line, whereas indirect estimation via the associated electric current drawn by the motor is marked by a black solid line. Finally, an alternative indirect measurement via the LC-derived longitudinal force is also plotted using a dashed black line. As seen in this figure, all three measurements show a good agreement. Similar results were observed on different surfaces.

Table 2.5.3.1: List of indirect signals.

Signal	Symbol	Sensors	Signal ID
Mechanical power	P_M	LC,JT	S12
Electrical power	P_E	LC	S13
Vertical force offset	d_x	LC	S14
Friction coefficient 1	μ_1	LC	S15
Friction coefficient 2	$P\mu_2$	LC	S16
Friction coefficient 3	μ_3	LC,JT	S17
Speed deviation	SD	JT	S18
Normalised speed deviation	SD_n	JT	S19
Sinkage	z	LC	S9

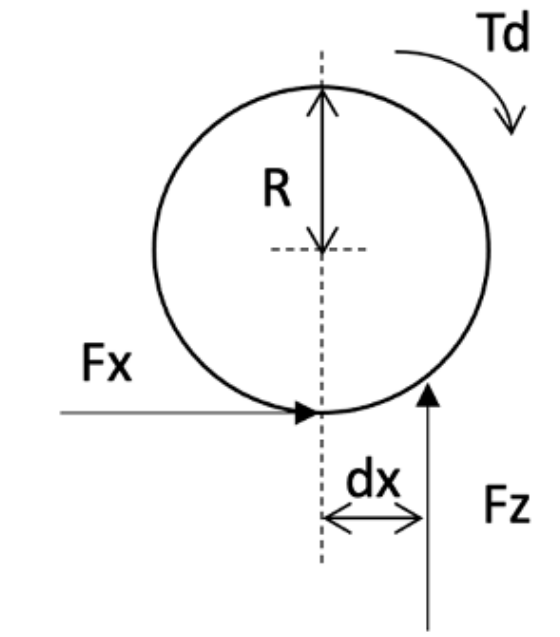


Figure 2.5.3.1: Definition of vertical force offset (dx).

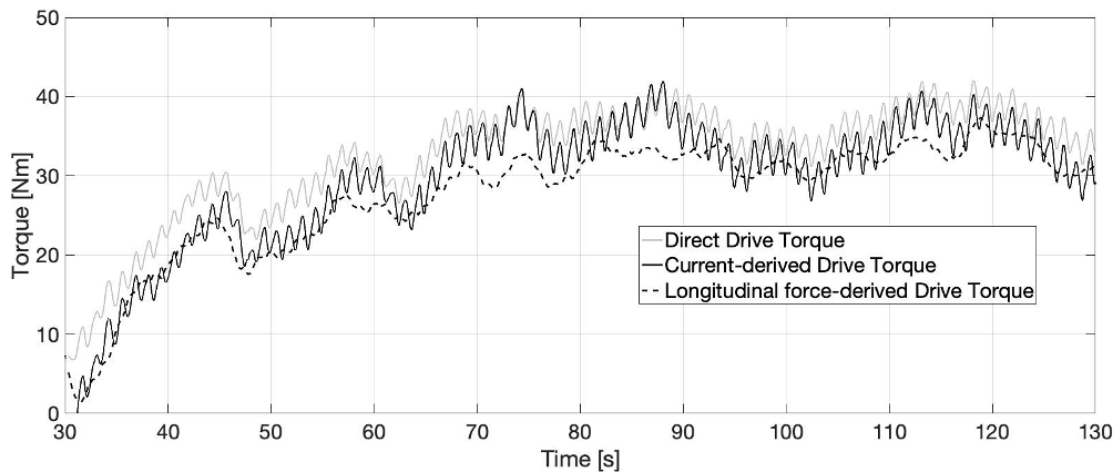


Figure 2.5.3.2: Torque applied by the left front wheel of SherpaTT as obtained from direct measurement of the load cell (solid grey line), indirect measurement via the electric current drawn by the motor (solid black line), or alternatively via the longitudinal force provided by the load cell (dashed black line).

2.5.4 Feature Extraction

First, each sensory signal is divided in time windows, and then, for each window features are extracted as the four main statistical moments. The size of the window, w_s is a design parameter. It is set as $w_s = 2$ s corresponding to a traversed terrain patch of about 20 cm (comparable with the wheel radius) at an average travel speed of 0.1 m/s. In previous works by the authors ([125]), it was found that this value of window size represents a good trade-off between informative content and spatial resolution. The four statistical moments are mean E , variance σ , skewness Sk and kurtosis Ku and are defined as follows:

$$\begin{aligned}
 E_i &= \frac{1}{N} \sum_{n=1}^N x_n \\
 \sigma_i^2 &= \frac{1}{N} \sum_{n=1}^N (x_n - E_i)^2 \\
 Sk_i &= \frac{1}{N} \frac{\sum_{n=1}^N (x_n - E_i)^3}{(\sqrt[3]{\sigma})^3} \\
 Ku_i &= \frac{1}{N} \frac{\sum_{n=1}^N (x_n - E_i)^4}{(\sqrt[4]{\sigma})^4}
 \end{aligned} \tag{2.5.4.1}$$

where x_n is the value of the signal at the n th time step and N is the total number of time-steps for the i th window. The extraction of the statistical features brings the size of the SVM-feature space to 80 (20 signals multiplied by their four statistical moments). The generic feature will be indicated as $SiMj$, where i ($i = 1, \dots, 20$) represents the signal ID, whereas j represents the statistical moment ($j = 1, \dots, 4$).

2.5.4.1 Feature Selection

Retaining only the features with the highest information content reduces the computational cost while preserving the accuracy of the model. The selection process can be performed via feature scoring using appropriate validity indices. Then, an iterative search algorithm can be followed to select a reduced best feature space.

Validity indexing A validity index can be assigned to each feature. This index represents a measure of the information content of the feature. In this study, two validity indices are considered: the Pearson Coefficient (PC) ([55]), and the WB index ([139]). The PC index can be computed through linear regression of a feature against the 3 classes of terrain, for example, sand, gravel, and paved ground. The higher the PC, the larger the information content of the feature. Although this index can be successfully used for 2-class classification problems ([32]), it might be difficult to implement it for multiclass cases like the one presented in this study, because the number assigned to each type of terrain is arbitrary. To overcome this issue, first, the PC index is computed

for each terrain pair (e.g., sand-gravel, gravel-paved ground, and sand-paved ground), and then averaged. For example, the PC index of the feature $SiMj$ against the Classes 1 and 2 (sand and gravel) can be calculated as ([53]):

$${}_2^1PC_{SiMj} = \frac{cov({}_2^1F_{SiMj}, {}_2^1y)}{\sqrt{var({}_2^1F_{SiMj}) var({}_2^1y)}} \quad (2.5.4.2)$$

where ${}_2^1PC_{SiMj}$ is a vector containing all values of the feature $SiMj$ for terrains 1 and 2, whereas ${}_2^1y$ contains class values (1 or 2) for each element of ${}_2^1F_{SiMj}$. Similarly, ${}_2^3PC_{SiMj}$ (PC index of feature $SiMj$ against the classes gravel and paved ground) and ${}_3^1PC_{SiMj}$ (PC index of feature $SiMj$ against the classes sand and paved ground) follow the same principle. The overall PC index for feature $SiMj$ can be now computed as follows:

$$PC_{SiMj} = \frac{{}_2^1PC_{SiMj} + {}_2^3PC_{SiMj} + {}_3^1PC_{SiMj}}{3} \quad (2.5.4.3)$$

In addition, the WB index can be computed for feature $SiMj$:

$$SB_{SiMj} = m \cdot \frac{SSW_{SiMj}}{SSB_{SiMj}} \quad (2.5.4.4)$$

where SSW is the sum of a square within classes and SSB is the sum of squares between classes, computed as follows:

$$\begin{aligned} SSW_{SiMj} &= \sum_{k=1}^{nCl} \sum_{s=1}^{n_k} (x_s - \mu_k)^2 \\ SSB_{SiMj} &= \sum_{k=1}^{nCl} n_k (\mu_k - \mu)^2 \end{aligned} \quad (2.5.4.5)$$

where x_s is the s th sample of feature $SiMj$, μ_k is the class k centroid value, μ is the overall data set centroid value, n_k is the number of samples in class k and nCl ($=3$) is the number of classes. A low value of WB_{SiMj} indicates that classes form compact and distant clusters relatively to feature $SiMj$. Therefore, the score assigned to each feature will be WB^{-1} : the higher the WB^{-1} , the better the feature for classification purposes. The rationale behind using two validity indices is that the WB and PC have two different statistical meanings: the former describes the compactness of classes, the latter shows the correlation between a given feature and the type of terrain. One may think that a feature with a low value of PC index will also have a relatively low value of WB^{-1} index. However, this is not always true, and exceptions do occur. For example, 2.5.4.1 shows the distribution of PC and WB indices for the 25 features with the highest scores. S6M2 is the feature with the third highest value of PC index, but it is only the 21st feature in terms of WB^{-1} . Similarly, S16M2 is the feature with the second-highest value of WB^{-1} index, but it is only the 14th in terms of PC. This shows that the two

indices rank the features in different ways, therefore they complement each other very well.

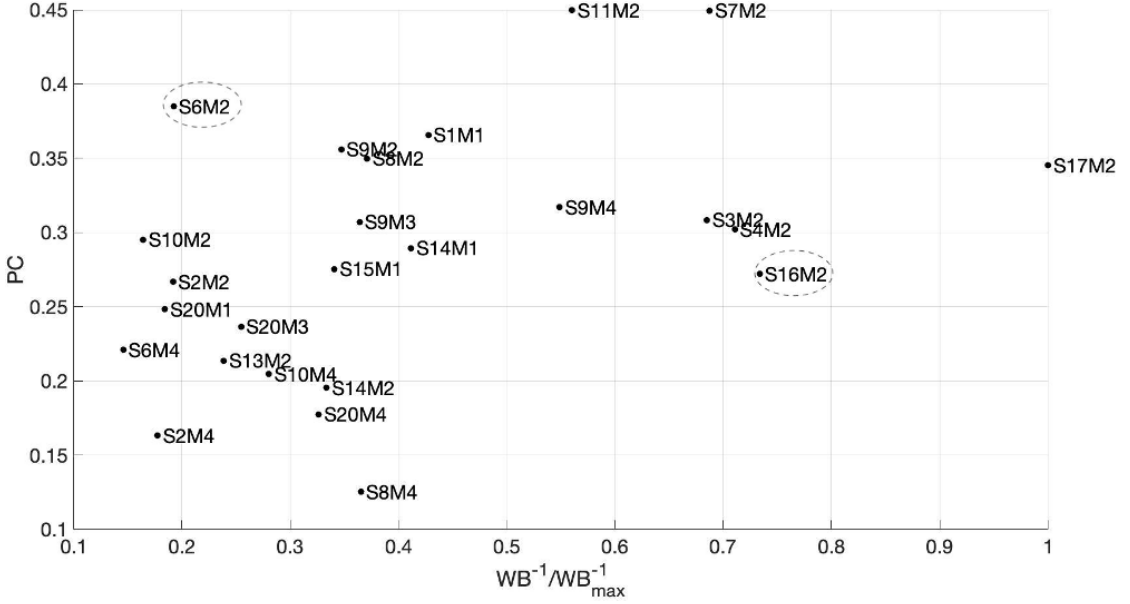


Figure 2.5.4.1: PC and WB indices distribution for the most relevant features.

Selection algorithm The proposed selection approach is based on the iterative search scheme presented in the block diagram of Figure 2.5.4.2. The input to the algorithm is the full set of n ($=80$) features. These features are then ranked using the output of one of the two validity indices (PC or WB) as a score. The best feature set is initialized with the first $n_{min} - 1$ ($=2$) features of the ranking. At this point, the objective is to iterate on all the remaining features to find those which provide better classification performance. In each iteration, identified with the index i that varies from n_{min} to n_{feat} , the i th feature in the ranking is added provisionally to the best feature set. Then, an SVM-based classifier is trained and evaluated in terms of F1-score via fivefold cross-validation. The k -fold cross-validation process partitions data into k randomly chosen subsets (or folds) of roughly equal size. Therefore, to improve the robustness of the feature selection algorithm, the training phase is repeated n_{train} ($=10$) times and the final F1-score is computed as the average of the scores obtained at each training phase. If the final F1-score is sufficiently higher than the best F1-score obtained so far, the i th feature is kept in the best feature set, and the best F1-score is updated. Otherwise, the i th feature is discarded from the best set and not considered for training purposes. To facilitate the reading of the block diagram in Figure 2.5.4.2, the meaning and the numerical values of the parameters involved in the selection process are collected in Table 2.5.4.1. The selection process discussed in Figure 2.5.4.2 can be repeated for each one of the two validity indices. Eventually, two best reduced feature spaces will be obtained: one associated with the PC and the other with the WB index.

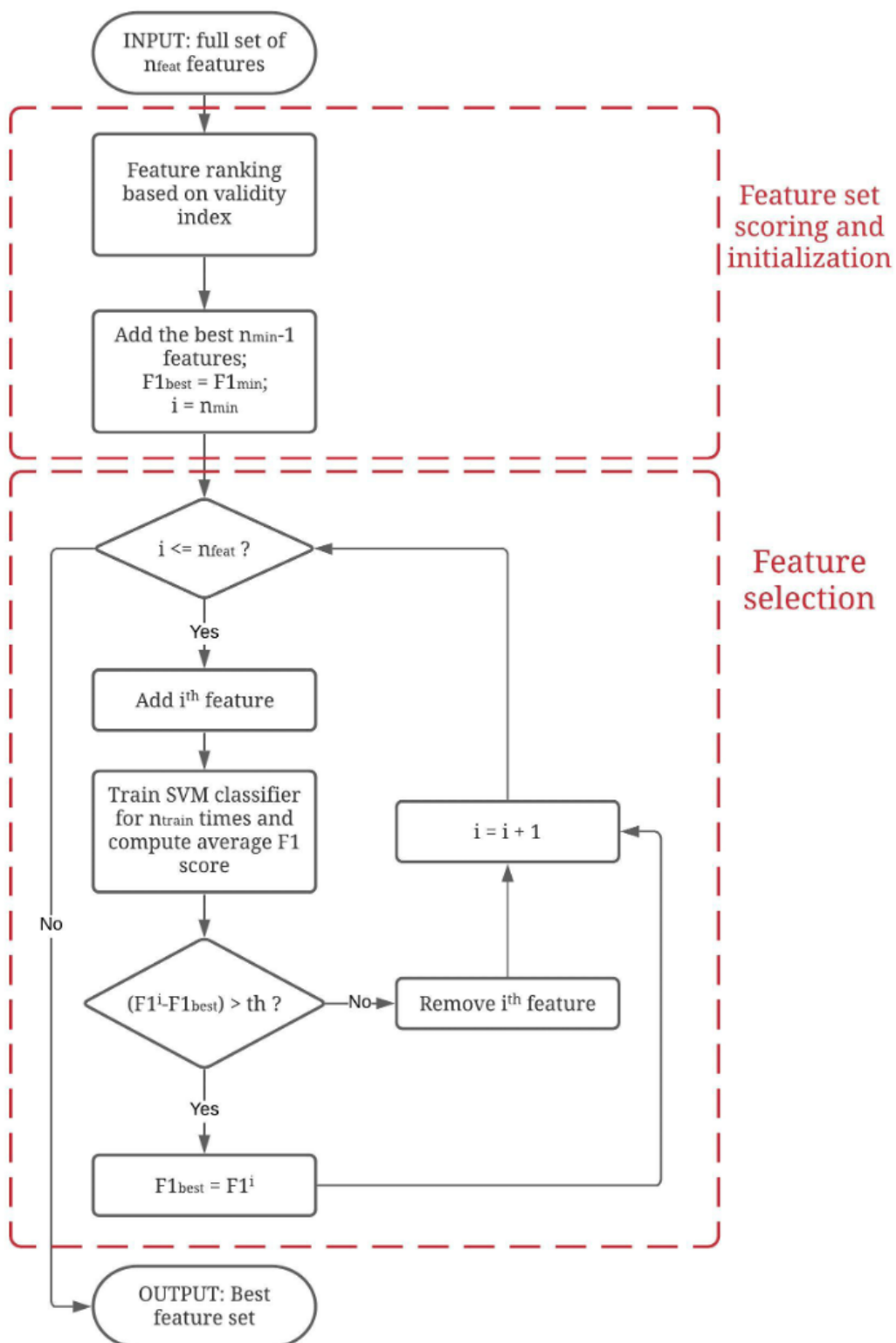


Figure 2.5.4.2: Block diagram of the proposed feature selection algorithm.

To further improve the robustness of the selection algorithm, the union of these two sets is chosen as the best for SVM training purposes. The 18 selected features are listed in Table 2.5.4.2. It is worth noting that three features extracted from indirect signals are included as well, thus, proving the utility of the signal augmentation phase.

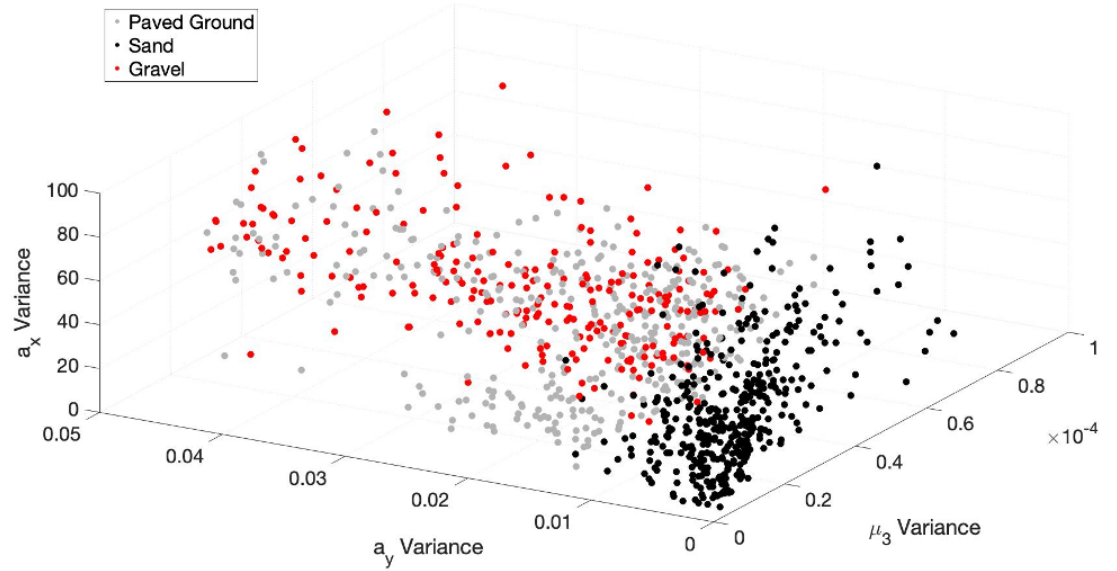


Figure 2.5.4.3: PC an WB indices distribution for the most relevant features.

A 3D plot of the three most relevant features in terms of WB index is shown in Figure 2.5.4.3 to help the reader to easily visualize the result of the whole selection process. As shown in this figure, the sand data form a quite compact cluster, with relatively low values of all three features. Conversely, gravel and paved ground data show higher values of S7M2 (variance of a_y) than sand and differentiate prevalently for values of S17M2 (variance of μ_3).

Table 2.5.4.1: List of parameters involved in the feature selection approach

Parameter	Description	Value
n_{min}	Minimum number of features	3
$F1_{min}$	Minimum F1-score	60%
th	Accepted improvement (threshold) in the F1-score	50%
n_{train}	Number of trainings for each new best feature set	5
n_{feat}	Number of features in the initial full feature set	80

Table 2.5.4.2: Best feature set

Signal	Statistical moment	Direct or Indirect	Feature ID	$\frac{WB^{-1}}{WB_{MAX}^{-1}}$	PC
μ_3	Variance	Indirect	S17M2	1.00	0.345
a_y	Variance	Direct	S7M2	0.691	0.449
T_d	Variance	Direct	S3M2	0.685	0.308
$gyro_z$	Variance	Direct	S11M2	0.561	0.450
$gyro_x$	Kurtosis	Direct	S9M4	0.549	0.317
F_x	Mean	Direct	S1M1	0.428	0.366
a_z	Variance	Direct	S8M2	0.371	0.350
$gyro_x$	Skewness	Direct	S9M3	0.364	0.307
$gyro_x$	Variance	Direct	S9M2	0.348	0.356
μ_1	Mean	Indirect	S15M1	0.342	0.275
z	Kurtosis	Indirect	S20M4	0.327	0.177
a_x	Variance	Direct	S6M2	0.192	0.385
F_z	Variance	Direct	S2M2	0.192	0.267
$gyro_y$	Variance	Direct	S10M2	0.164	0.295
F_x	Variance	Direct	S1M2	0.143	0.253
a_y	Kurtosis	Direct	S7M4	0.048	0.115
PWM_d	Mean	Direct	S5M1	0.013	0.062
$gyro_z$	Mean	Direct	S11M1	0.011	0.064

2.5.5 Experimental Results

In this section, the results of the generalization problem are shown on the main data set. Next, results for two extrapolation cases are presented.

2.5.5.1 Generalization

In the generalization problem, only the main data set is used (e.g., experiments on paved ground, gravel, and sand). The algorithms are tested via fivefold cross-validation. The data set comprises of 1204 samples, where a sample corresponds to a 2-second time window. Of these 1204 samples, 443 are collected on paved ground, 338 on gravel, and 423 on sand.

One of the objectives of this research is to demonstrate how a proper feature selection algorithm can reduce the computational and memory cost of the model, while maintaining a similar accuracy in prediction. Table 7 shows a comparison between the two machine learning algorithms in terms of accuracy and computational burden. Moreover, SVM is tested with three different feature sets:

- Direct feature set (44 features);
- Full feature set (80 features);

- Best feature set (18 features).

while CNN is tested with three different signal sets:

- Direct signal set (11 signals);
- Full signal set (20 signals);
- Best signal set (13 signals).

The signals used for training CNN correspond to those used to compute SVM features. In fact, the 44 direct features are the four statistical moments of the 11 direct signals, and the full 80-feature set is composed of the four statistical moments of the full 20-signal set. Furthermore, the training set for CNN includes the signals used to derive the features in the best feature set. Namely, the 13 best signals are: friction coefficients 1 and 3, longitudinal, lateral, and vertical accelerations, drive torque, yaw, pitch and roll rates, longitudinal and vertical forces, sinkage, drive PWM.

The accuracy of the SVM model trained with the direct and full feature sets is 89.8% and 90.8%, respectively. With the full feature set, more samples are correctly classified by SVM, but memory usage has increased by 82%, training time by 32%, testing time by 71%, and feature extraction time by 50%. This proves the effectiveness of the signal augmentation in terms of accuracy and shows the drawbacks in terms of the computational burden. The purpose of feature selection is to reduce the computational cost, without losing classification accuracy. The results presented for SVM trained with the best feature set, prove that the feature selection algorithm proposed in this study is effective. In fact, the accuracy reaches 90.9% and when compared to the SVM trained on the full feature set, while the model memory usage is reduced by 77%, training time by 6%, testing time by 29%, feature extraction time by 33%.

The effectiveness of both input signal augmentation and feature selection is also confirmed by the results presented for CNN. This deep learning algorithm gains in terms of accuracy from signal augmentation reaching 96.4%. Using the full signal set still results for CNN in the same drawbacks presented for SVM: model memory usage increased by 18%, training time by 36%, feature extraction time by 77%. In contrast with SVM, testing time for CNN with full signal set is reduced by 22%. Training CNN with the best signals resulting from feature selection leads to an accuracy of 96.2% and when compared to the full-signal CNN, the model memory usage is reduced by 14%, training time by 27%, testing time by 4%, feature extraction time by 41%.

Feature extraction times presented in the last row of Table 2.5.5.1 are suitable for online applications for both SVM and CNN, even if construction of multichannel spectrograms from best signals for CNN takes about 2.1 ms more than the construction of best features for SVM. It should also be noted that feature extraction time for both SVM and CNN can be further improved by optimizing the current MatLab code using vectorization or processing the data directly with a C++ code.

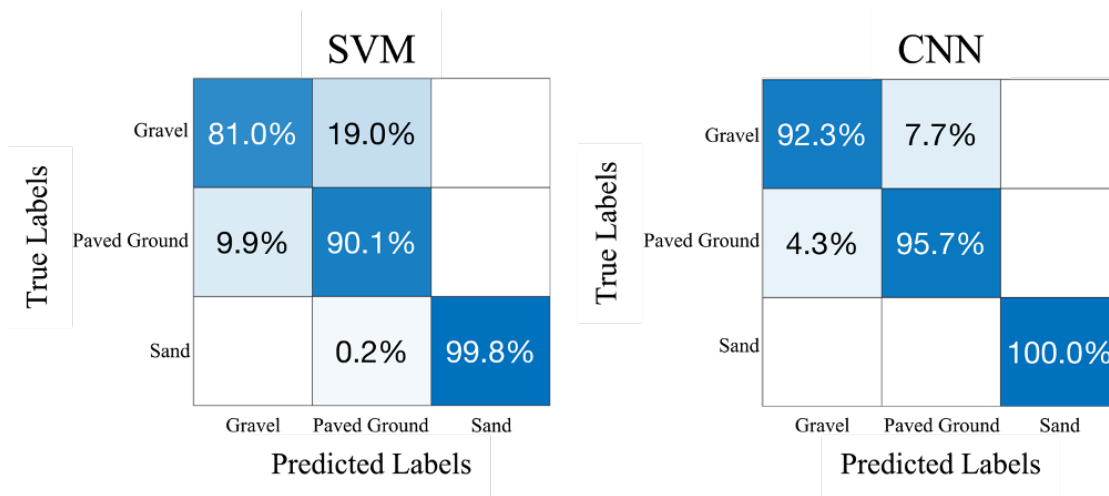


Figure 2.5.5.1: Generalization results for best features SVM and best signals CNN. CNN, convolutional neural network; SVM, support vector machine.

Confusion matrixes for both SVM and CNN are shown in Figure 2.5.5.1 only for best feature and best signal sets. Sensitivity results for each class are contained in the diagonal elements of each confusion matrix. The performance of both models in terms of precision, recall, and F1-score are shown in Table 2.5.5.2.

Table 2.5.5.1: Performance comparison between terrain classifiers trained on different feature sets: direct, full, best feature set

Feature and signal sets	SVM			CNN		
	Direct	Full	Best	Direct	Full	Best
Accuracy (%)	89.8	90.8	90.9	95.6	96.4	96.2
Model Memory usage (kB)	547.6	996.9	228.0	44.9	53.2	45.8
Training time (ms)	118.9	157.7	148.0	1.07e4	1.46e4	1.06e4
Testing time (ms)	17.4	29.8	21.2	153.0	119.4	114.7
Feature extraction time (ms)	0.6	0.9	0.6	2.6	4.6	2.7

Table 2.5.5.2: Accuracy, Precision, Recall and F1-score for SVM and CNN in generalization

Class	SVM			CNN		
	Gravel	Paved ground	Sand	Gravel	Paved ground	Sand
Precision (%)	89.1	82.4	100	80.3	82.2	100
Recall (%)	81.0	90.1	99.8	92.3	95.7	100
F1-score (%)	84.9	86.1	99.9	85.9	88.4	100

Both models perform good in the generalization of data, with CNN being slower but significantly more accurate. This increase in classification accuracy is not the main advantage for CNN classification model with respect to SVM. Where the two models show the greatest difference in classification performance is indeed extrapolation, as shown in the next section.

2.5.5.2 Extrapolation

In the extrapolation problem, the operating conditions of training and testing sets are different, therefore these sets do not come from the same population. In this study two extrapolation cases are presented. The first one deals with varying rover speed, whereas the second one assesses the performance of the algorithms on a terrain unseen in the training phase.

Testing on a new vehicle velocity During the experiments with SherpaTT, the rover was controlled at two different speeds: 0.1 m/s and 0.15 m/s. Of the 14 runs, 7 were conducted at low speed (0.1 m/s) and 7 at high speed (0.15 m/s). Data collected at low-speed form the low-speed distribution, whereas data collected at high-speed belong to the high-speed distribution. In the extrapolation problem presented here, low-speed data are used as training set, while high-speed data are used as testing set. Both sets belong to the main data set (paved ground, gravel, and sand). Proprioceptive sensorial data are very useful for terrain classification but also show a strong dependency from traversing speed ([6]). Most terrain classification algorithms analyse and classify proprioceptive data acquired at constant traversing velocity on different terrains. Studies have been also conducted to show the dependency of terrain classification performances from rover's traversing speed, searching for the velocity that maximizes classification performance. For being able to classify the traversed terrain at any travelling speed a rover should be equipped with a model trained on a vast variety of possible traversing speeds or could only use speed-independent features that are difficult to construct and may not be well suited for terrain classification. Another way of achieving the goal of sensing and classifying the terrain at any travelling speed is using a model that shows good results when tested on data acquired at a traversing velocity different from the one used for training. Figure 2.5.5.2 contains the confusion matrixes for both SVM and CNN when trained on low-speed data and tested on high-speed ones. As can be seen, despite both models showed good results in generalization only CNN is also capable of extrapolating the information of the traversed terrain from data acquired at a different speed. The two models were still trained and tested using only best feature set for SVM and corresponding signal set for CNN. While CNN keeps classification accuracy as high as 89.5%, SVM becomes unreliable achieving only 55.7% of correctly classified data samples. The performances of both models in terms of precision, recall, and F1-score are shown in Table 2.5.5.3. It should also be pointed out that high-speed data used as testing constitute 50% of available data, representing, therefore, testing set larger than the one usually used (20%-30%). The robustness of CNN's classification performance on a large testing set composed of data acquired at a different speed suggests that this model is well

suited for terrain classification purposes. Moreover, the features automatically learned from signal spectrograms appear to be more reliable than statistic ones and represent a better choice to be able to classify the traversed terrain at various traveling velocities. Similar results are obtained when trained on high-speed data and tested on low-speed data, and they are omitted for brevity sake.

Table 2.5.5.3: Precision, Recall and F1-score for SVM and CNN in extrapolation using varying velocity

Class	SVM			CNN		
	Gravel	Paved ground	Sand	Gravel	Paved ground	Sand
Precision (%)	54.8	46.8	100	80.3	82.2	100
Recall (%)	40.3	83.9	47.5	82.1	81.0	99.5
F1-score (%)	46.4	60.1	64.4	81.2	81.6	99.7

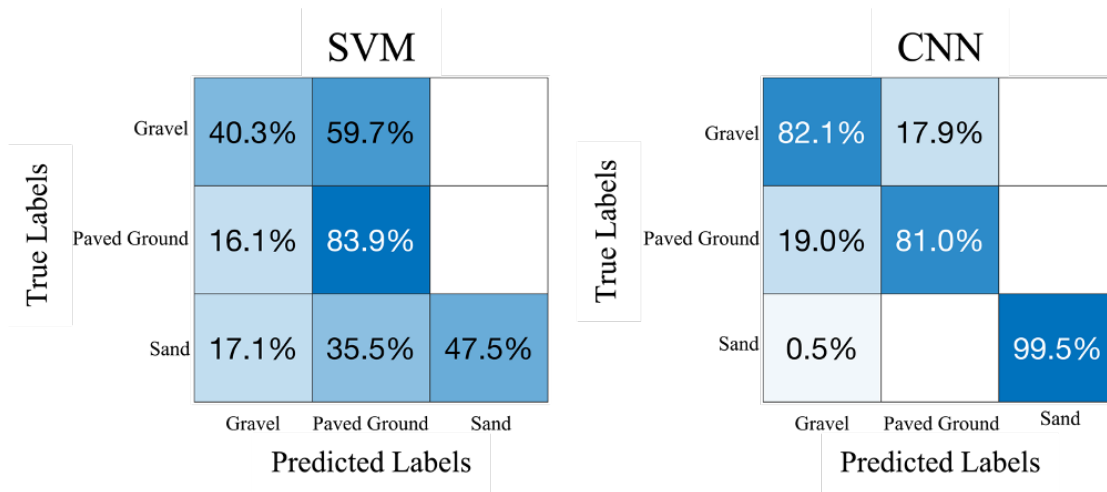


Figure 2.5.5.2: Extrapolation results for best features SVM and corresponding signals CNN.

Testing on an independent data set The second extrapolation use case aims to evaluate the system response when labeling observations collected on a terrain different from those used in training (independent data set). To this aim, the ground classifier previously trained on the main data set (formed by paved ground, gravel, and sand) is further validated on a representative data set gathered from a second field test campaign run in a planetary analogue terrain in a sand mine near Bremen (see Figure 2.5.5.3).

For this extrapolation challenge, we have tried to generalize the classification problem at hand by referring to terrain difficulty labels rather than specific terrain classes, as explained in Table 10. Adopting the proposed terrain difficulty scale, paved ground and

sand can be seen as the opposite extremes. Firm-ground offers better traction and less compressibility, therefore a low-difficulty label can be assigned to it. Conversely, soft ground poses more challenges, and it is scored as a highly difficult surface. Then, the difficulty degree associated with an unknown observation can be considered as inversely proportional to the distance from the class sand. One should note that such a generalization effort can be useful or necessary for the practical implementation of planetary exploration terrain classifiers that can be only trained on Earth using representative analogue surfaces, and then applied to unknown planetary surfaces via extrapolation.



Figure 2.5.5.3: (a) Sherpa TT during the sand mine testing; (b) a close up of the tracks left by the wheels.

The sand mine independent data set consists of 302 samples, where, again, a sample corresponds to a 2-second window. It should be also underlined that, although ground-truth data is not available for this extrapolation problem, the terrain in the sand mine can be expected as a surface with medium-high difficulties, like the sand type of the main data set (Figure 2.3.3.1) but somewhat more compact and humid. As an indicative measure, sample tracks left by the wheels on the sand mine terrain are shown in Figure 2.5.5.3(b).

Table 2.5.5.4: Category of difficulty assigned to each terrain type of the training set and predictions as obtained from SVM and CNN in the sand mine test

Terrain type	Equivalent category of terrain difficulty	SVM	CNN
Sand	High	215	211
Gravel	Medium	52	73
Paved Ground	Low	35	18

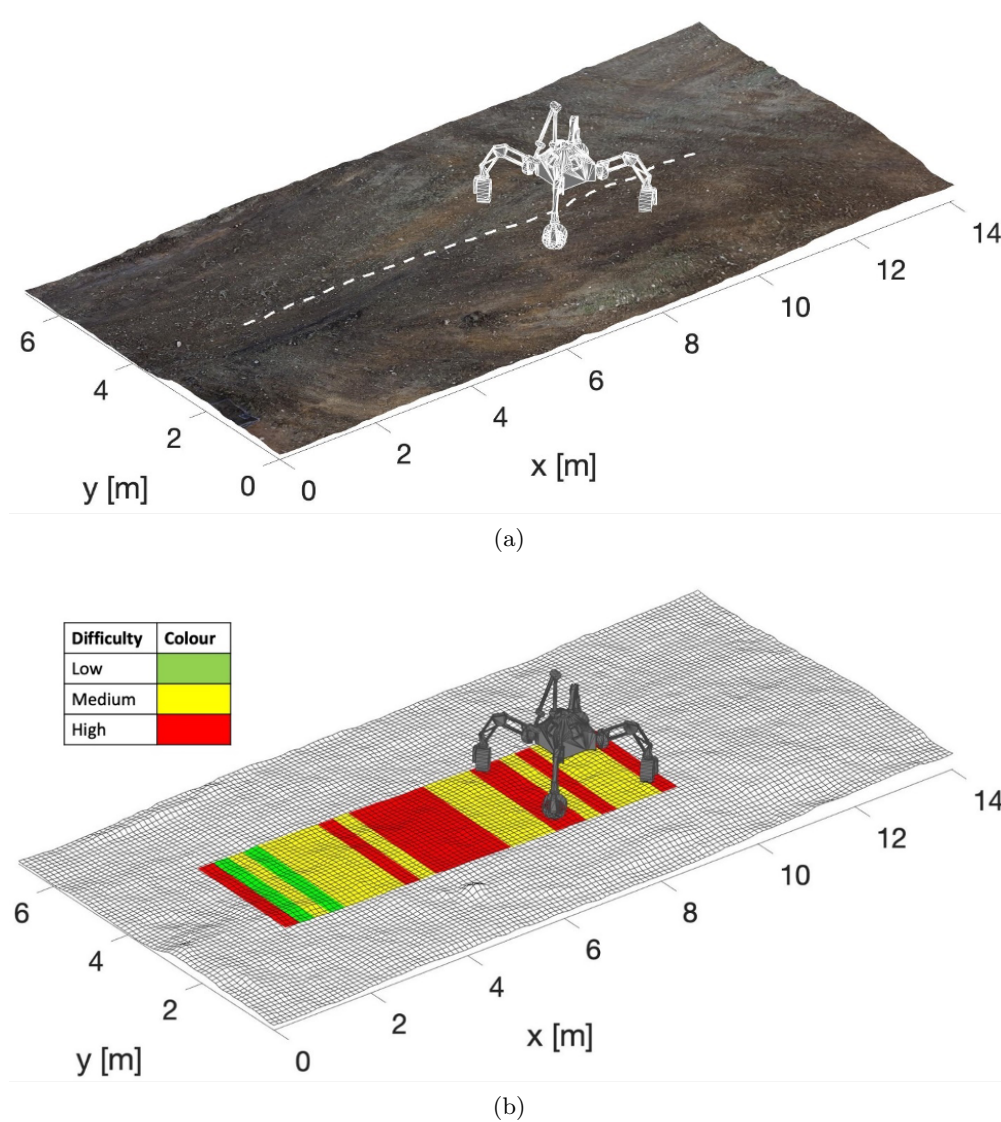


Figure 2.5.5.4: Semantic labeling using discrete terrain difficulty categories: (a) 3D stereo-generated map of the environment with overlaid the path (dashed white line) followed by the rover, (b) corresponding terrain difficulty visualization. Terrain patches are marked respectively in red, yellow, and green, for high, medium, and low difficulty.

The classification results obtained from SVM and CNN are also collected in Table 2.5.5.4 showing predicted labels of terrain difficulty. Out of the 302 samples, the SVM-based algorithm classifies 71.2% as high difficult terrain, 17.2% as medium and 11.6% as low. CNN performs similarly, classifying 69.9% of the new terrain samples as highly difficult, 24.2% as medium, and 5.9% as low. A relatively low percentage of the test samples (about 12% for SVM and 6% for CNN) is classified as hard soil. For easier

visualization, the results obtained from the CNN-based classifier are presented in Figure 2.5.5.4 during a sample straight run using semantic labelling where the successive terrain patches traversed by the rover are marked according to a color map that reflects the terrain difficulty scale of Table 2.5.5.4 (see also to the inset of Figure 2.5.5.4 (b)). We recall that three discrete levels of terrain difficulty are considered: low, medium, and high.

Figure 2.5.5.4 (a) shows the 3D stereo-generated map of the environment with overlaid a CAD model of SherpaTT and the path followed by the rover denoted with a dashed white line, whereas in Figure 2.5.5.4 (b) the corresponding terrain labeling is reported with terrain patches marked respectively in red, yellow, and green, for high, medium, and low difficulty. In this test that was performed on fairly homogeneous terrain, the system mostly classifies the sand mine surface as of medium-high difficulty with two erroneous predictions (low difficulty) between 2 and 3 m.

Considerations This study presented an approach to soil classification that relies on proprioceptive sensing only, for example, accelerations, forces, torques, and electrical currents. The algorithms developed are validated on data collected during tests performed with the hybrid wheeled-legged rover SherpaTT. The physics-based signal augmentation process presented in this research uses 11 proprioceptive measurements to produce a large set of 80 features for SVM and 20 signals for CNN. This improved the information content as proved by the high classification accuracy obtained in generalization (90.8% for SVM and 96.4% for CNN). The proposed feature selection algorithm allows SVM to retain a high classification accuracy with only a portion of the full set (18 features), with successful reductions in memory usage (-77%) and required time for training (-6%), testing (-29%), and feature extraction (-33%). The same benefits also apply for CNN when using a reduced set of 13 signals related to the 18 best SVM features, improving memory usage (-14%), training time (-27%), testing time (-4%), and feature extraction time (-41%). The comparison between SVM and CNN shows good capabilities of both models in generalization, with accuracy higher than 90%. More challenging extrapolation problems have been tackled as well to evaluate the impact of varying operating conditions and site of the acquisition. In these tests, CNN outperformed the SVM counterpart. When tested on a new vehicle velocity, CNN reached an accuracy of 89.5%, against 55.7% held by SVM. When tested on new terrain, CNN recognized its deformability class more frequently than SVM, correctly classifying 6% more of the available samples. Based on these results, the proposed CNN qualifies as a good algorithm for soil classification even in the presence of disturbances and unknown conditions. This study proved that is possible to use only proprioceptive features to infer the signature of a particular surface via learning algorithms. Moreover, the presented promising results suggest the possibility to extend rover traveling distance thanks to on-board integration of the developed learning algorithms.

Future developments of this study refer to (i) continuous training of the system by incorporating instances of “new terrain” classes during normal operations, therefore making the system adaptive, (ii) augmenting the classifier with new special classes; for

example, instances of excessive wheel slippage (close to 100%) can be used to train a hazard class to inform the rover of impending immobilization conditions, (iii) combining the proposed framework using proprioceptive signals with exteroceptive signals. The latter would enable the vehicle to predict hazards or trapping conditions before driving through the ground, for example, based on noncontact information coming from vision sensors.

Chapter 3

Prediction of Vehicle-Terrain interaction

3.1 Motion Resistance Prediction

This chapter of the thesis is based on the published paper [121]. Wheel-terrain interaction plays a critical role for vehicle mobility on natural terrain, such as in agricultural, planetary exploration and off-road settings. Estimation of the terrain characteristics and the way they affect traversability is essential for the vehicle to better plan its safest and energy-efficient path. This work proposes a novel approach to learn and predict from a distance the motion resistance encountered by a robotic vehicle, while traversing natural soil, by using visual information from a stereovision device. To this end, terrain appearance and geometry information are first correlated to resistance torque measurements during a learning phase via two alternative regression approaches, namely Least-Squares Boosting and Long-Short Term Memory Recurrent Neural Network. Then, such a relationship is exploited to predict motion resistance remotely, based on visual data only. Results obtained in preliminary experimental tests on ploughed and compact terrain are presented to show the feasibility of the proposed method.

On natural terrain, as in agricultural and off-road settings, the mobility of a vehicle is known to be highly influenced by wheel-terrain interaction and can be very different on ploughed terrain rather than on dirt road or compacted soil [10]. The knowledge of the terrain characteristics and its ability to support vehicular motion can be beneficial in many ways. Locomotion performance can be optimised in terms of traction or power consumption (e.g., fuel or battery life) by adapting control and planning strategy to site-specific environments. Terrain estimation can also contribute to increase the safety of agricultural vehicles during operations near ditches, on hillsides and cross slopes, as well as on highly-deformable terrain [128]. Another important aspect that is raising interest in precision agriculture is related to the prediction of the risk of soil compaction by farm machinery [118]. Early work on vehicle-terrain interaction estimation mainly relied on model or expert rule-based observers [107],[93]. More recently machine learning approaches have been developed as an alternative or complementary solution with a

focus on slip estimation [5],[111].

This research presents a novel framework for learning and predicting online the motion resistance of an agricultural robot on natural terrain. The main novelty of the proposed approach relies on the use of visual information, including appearance and geometric characteristics of the ground, to learn and predict from a distance the resistance torque encountered by the vehicle when traversing different terrain types. Estimation of the expected torque before entering a terrain is crucial for the vehicle to better plan its safest path and avoid dangerous or difficult areas. To address this problem, terrain appearance and geometry information are correlated to the torque measured by the rover while traversing a given surface. Such relationship is learned from previous experience (learning phase), so that, successively, motion resistance can be predicted from visual information only (prediction phase).

Section 3.2 presents the rover used as test bed with its sensor suite as well as an overview of the proposed algorithms. Section 3.3 discusses the experimental results obtained.

3.2 Husky rover application

The Husky A200 robotic platform was used for experimental sensor data gathering (Figure 3.2.0.1). The sensor suite includes electrical current and voltage sensors to measure wheel mechanical torque and a colour stereocamera (Point Grey XB3) mounted on a dedicated aluminium frame in a looking-forward configuration, which provides three-dimensional reconstruction of the environment with RGB colour data.

The proposed approach aims at learning and predicting the motion resistance encountered by a vehicle on a previously unknown terrain from visual data. During the learning phase, the rover traverses the terrain and collects visual information (appearance and geometry) about a future location. When this location is reached by the rover, a motion resistance measure is taken using onboard current sensors. Then, a mapping function between visual and motion resistance information is learnt. After learning, the expected torque can be predicted based on the learnt functional relationship using only stereo imagery as input. Visual information is represented in terms of colour and geometric features. Specifically, for a given terrain patch, a feature vector is defined using mean, variance, skewness and kurtosis of each channel of the $c1c2c3$ colour space. For the same patch, geometric features are extracted based on the power spectral density (PSD) of the surface profile [110]. Two approaches for regression fitting and prediction are compared, namely Least-Squares Boosting (LSBoost) and Long-Short Term Memory Recurrent Neural Network (LSTM-RNN). LSBoost is an ensemble machine learning algorithm. At every step, the ensemble fits a new learner to minimize the mean-squared error between the observed response and the aggregated prediction of all learners grown previously [17]. In the learning phase, a Bayesian optimization was run to optimize the hyper-parameters of the boosted regression ensemble. The idea behind the ensemble algorithm is to meld results from many weak learners into one high-quality ensemble predictor. The weak learners used in this work are decision trees. The hyper-parameters

optimized during the Bayesian process are shown in Table 3.2.0.1.



Figure 3.2.0.1: Robotic platform used for data gathering.

Table 3.2.0.1: Hyper-parameters of the LSBoost model.

Hyper-parameter	Description	Range
NumLearningCycles	Number of ensemble learning cycles. At every learning cycle, the software trains one weak learner	[10,500]
LearnRate	Learning rate for LSBoost shrinkage	(0,1]
MinLeafSize	Minimum number of leaf node observations. Each leaf has at least MinLeafSize observations per tree leaf	[1,NumObservations/2]
MaxNumSplits	Maximal number of decision splits (or branch nodes)	[1,NumObservations-1]
NumVariablesToSample	Number of predictors to select at random for each split	[1,NumPredictors]

The Bayesian optimization algorithm attempts to minimize a scalar objective function $f(x)$, for x in a bounded domain, where x contains all the parameters to optimize. The key elements in the minimization are a model of $f(x)$ and an update procedure at each evaluation of $f(x)$. In this work, a Gaussian process model of $f(x)$ was used to obtain a posterior distribution after every evaluation of the objective function. The first point x_1 for the evaluation of $f(x)$ is chosen randomly, whilst for all the following iterations the evaluation point x_i is chosen by maximizing an acquisition function. More details about Bayesian optimization can be found in [41]. A Long-Short Term Memory Recurrent Neural Network (LSTM-RNN) was also implemented for comparison. The structure of a Recurrent Neural Network for the regression purpose is shown in Figure 2.4.1.1. A certain number of Recurrent Units (RUs) passes a hidden output h_i (with $i = 0, \dots, T - 1$, where $T + 1$ is the number of RUs) to the following RU and gives a guess of the torque value based on the instantaneous input and the previous hidden output[56]. The LSTM is a particular type of RU that has been shown to be capable of retaining through time in the hidden output relevant information from the input sequence to improve the accuracy of the prediction. The hyper-parameters of the training stage, as well as the parameters of the net architecture were manually tuned for each terrain observing the progress during training of RMSE and loss. They are presented in Section 3.3.

3.3 Experimental Results

The system was tested in a rural environment where two main surfaces were present (Figure 3.3.0.1):

- ploughed terrain: vineyard terrain broken and turned over;
- compact terrain: unbroken agricultural land. It is a compact and relatively hard terrain that can be typically found in olive groves.

The rover was remotely controlled to follow a straight line of about 67 m, driving parallel to a vineyard edge on ploughed terrain and of about 51 m on compact soil in an olive grove. In each test, 1145 samples of ploughed terrain and 872 samples of compact terrain were collected.

Table 3.3.0.1: Optimized LSBoost ensemble hyper-parameters.

Hyper-parameter	Optimization on ploughed terrain	Optimization on compact terrain
NumLearningCycles	491	70
LearnRate	0.09382	0.15418
MinLeafSize	2	5
MaxNumSplits	428	539
NumVariablesToSample	4	3

Table 3.3.0.2: Hyper-parameters of the LSTM-RNN model.

Hyper-parameter	Ploughed	Compact
Number of RUs	50	50
Sequence Length	50	10
Sequence Overlap	25	5
Validation Percentage	0.15	0.15
Initial Learning Rate	0.005	0.005
Learn Drop Factor	0	0
Learn Drop Period	10	10
Maximum Number of Epochs	1500	1500
Mini-Batch size	1	1
Validation Patience	10	10
Validation Frequency	50	50
Gradient Threshold	10	10

A sequential split of the data into training and test sets was performed. Specifically, for each terrain type, the first 70% of the sequence was used for training and the remaining part was used for testing. The Bayesian optimization for the LSBoost model was repeated for each terrain type and the best hyperparameters are shown in Table 3.3.0.1. Table 3.3.0.2 presents the hyperparameters of the LSTM-RNN model that have been manually tuned.

Table 3.3.0.3: Model errors.

Algorithm	LSBoost		LSTM-RNN	
Terrain	Ploughed	Compact	Ploughed	Compact
MAPE [%]	8.96	20.1	8.15	11.0
RMSE [Nm]	0.0573	0.0460	0.05221	0.0275

The first plot in Figure 3.3.0.2 (a) shows the prediction of the LSBoost ensemble model for ploughed terrain, compared with the observed values. The second plot in Figure 3.3.0.2 (a) shows the discrepancy between observations and predictions. Figure 3.3.0.2 (b) shows the same results for the compact terrain case. Similarly, Figure 3.3.0.3 (a) and (b) show the results obtained with the LSTM-RNN model on ploughed and compact terrain respectively. The performance of both models in terms of Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE) are shown in Table 3.3.0.3. It can be seen that LSBoost and LSTM-RNN perform similarly for the test on ploughed terrain leading to a MAPE of 8.96% and 8.15% respectively, whereas LSTM-RNN overcomes LSBoost performance in the case of compact terrain with a MAPE of 11.0% compared to the 20.0% obtained using the LSBoost model.

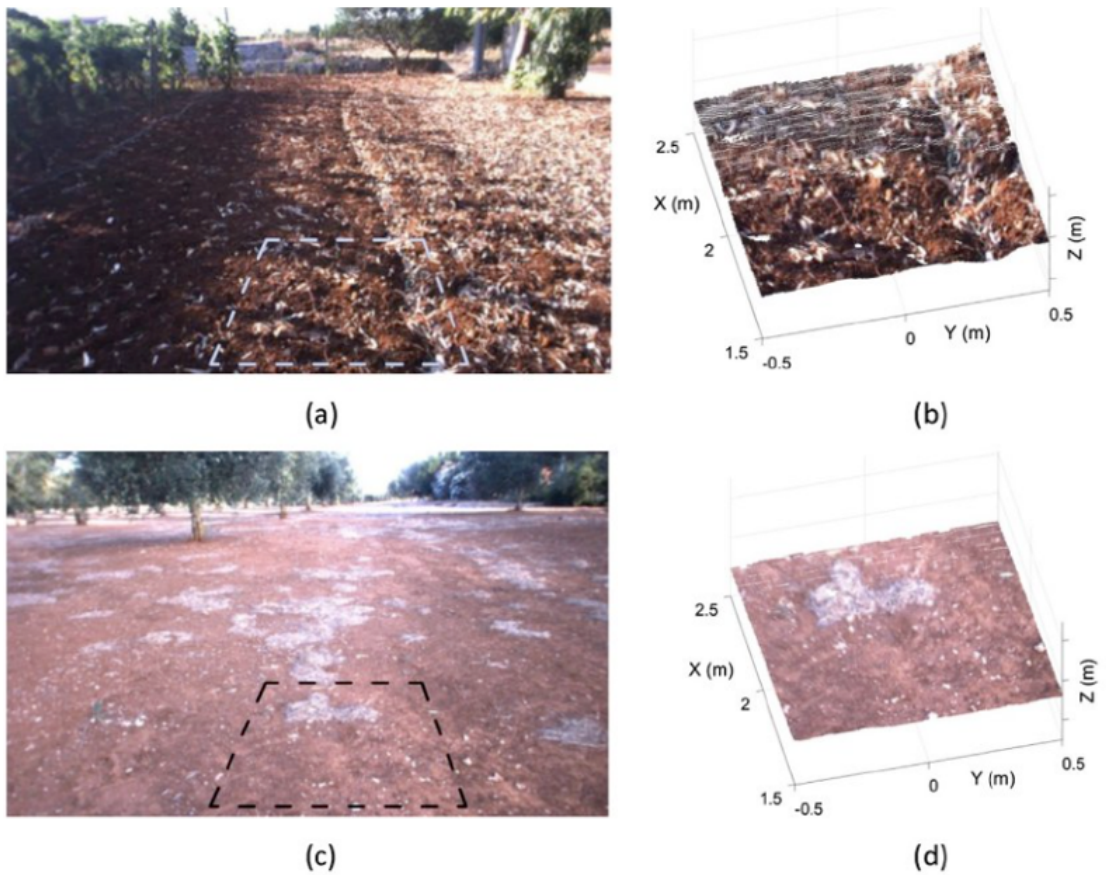
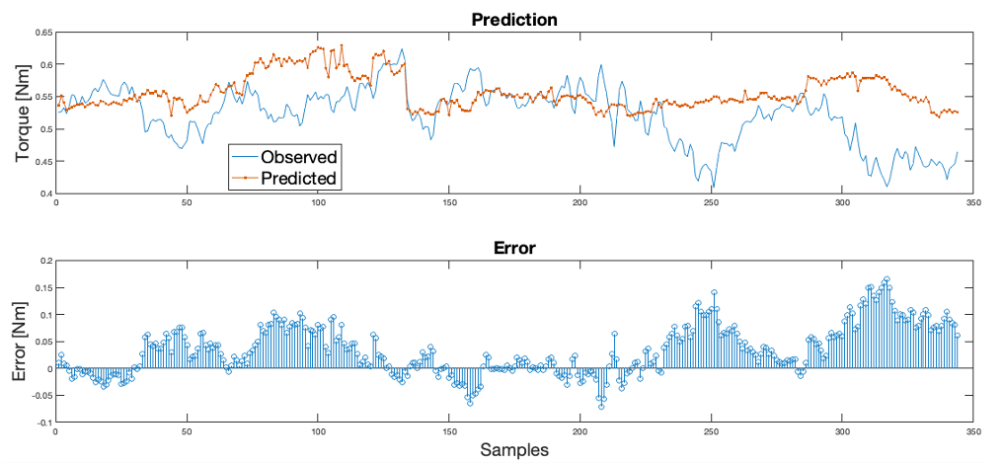
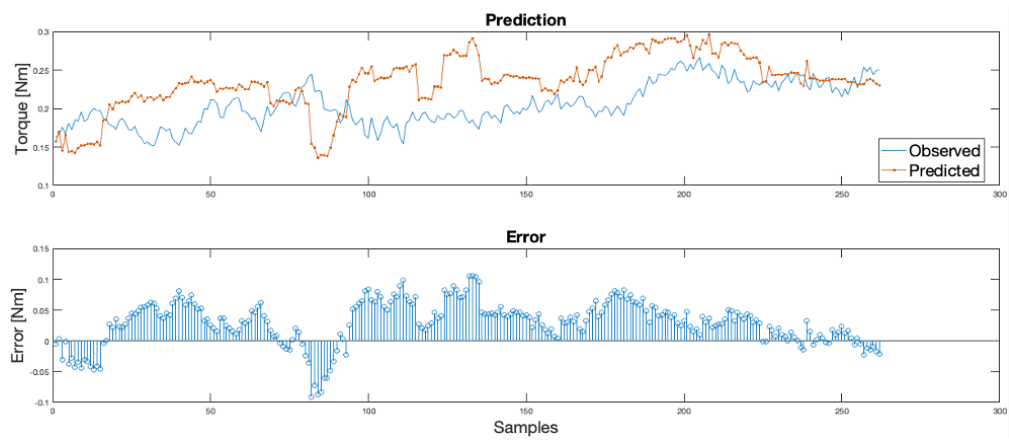


Figure 3.3.0.1: Terrain types: (a)-(b) ploughed agricultural terrain; (c)-(d) compact agricultural terrain. Left: sample images with overlaid the boundaries of the inspection window. Right: corresponding 3D colour terrain patches obtained from the stereocamera.



(a)



(b)

Figure 3.3.0.2: LSBoost predictions (a) for ploughed terrain and (b) for compact terrain.

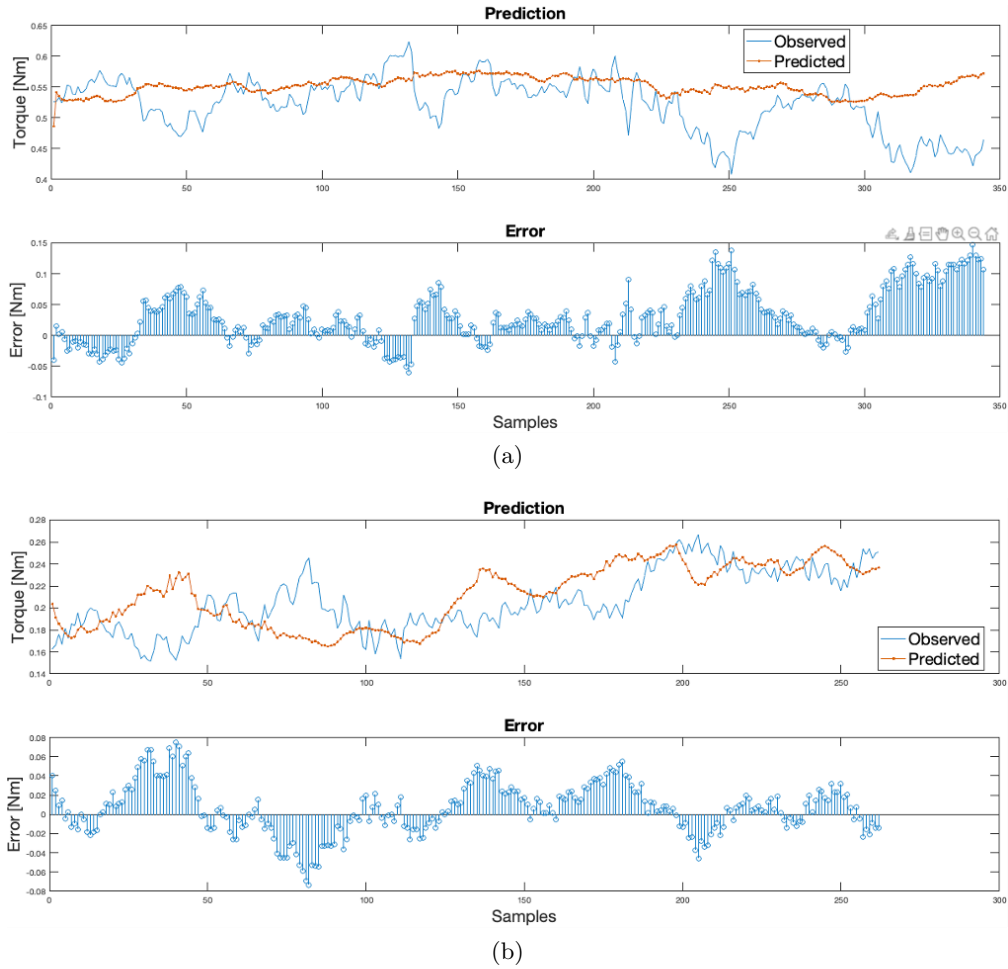


Figure 3.3.0.3: LSTM-RNN predictions (a) for ploughed terrain and (b) for compact terrain.

Considerations This research presented a novel learning and prediction framework to estimate remotely the motion resistance the vehicle will encounter on different types of soils based on visual information from an onboard stereovision device, mounted in a forward-looking setup. Specifically, first, the vehicle learns a mapping between visual terrain appearance and geometry and motion resistance based on a regression learner, then based on such a relationship it is able to predict the motion resistance of future locations based on visual information only. Two regression approaches have been implemented for comparison: a Least-Squares Boosting algorithm and a Long-Short Time Memory Recurrent Neural Network. Preliminary tests were performed on two terrain types showing that the system is able to make predictions with an error, in the worst case, of 20% for the LSBoost model and 11% for the LSTM-RNN.

Chapter 4

Multi-sensor Perception

4.1 Crop Monitoring Application

The availability of up-to-date and accurate data is an essential prerequisite for precision farming tasks, such as variable rate application of fertilizers/pesticides, identification of infected plants or invasive species, and controlled traffic farming. While satellite and airborne technologies have been in use for some decades to effectively provide multi-spectral and 3D information in wide agricultural and forestry areas, these platforms generally lack the resolution needed to observe stems, leaves or fruits. Satellite images typically have pixel resolution of hundreds of meters and airborne sensing may provide resolution of a few meters, whereas monitoring orchards or vineyards requires observations at a smaller scale. Information update frequency is also limited, varying from hours to several days. In crops with smaller extension, UAVs equipped with RGB, multi spectral or LiDAR sensors, have been adopted to overcome these bottlenecks, allowing for efficient crop survey at user-defined spatio-temporal resolutions to assess vegetation vigor or for canopy characterization [81]; [23]. However, in high density crops, using aerial data can still be ineffective for precise measurement at leaf/fruit level, e.g., for health status assessment and yield estimation.

As an alternative or complementary approach, proximal sensing from ground-based or manually deployed devices can be performed. Proximal sensors range from RGB cameras to high-resolution hyper spectral imaging, infrared (IR) thermal cameras, and 2D/3D LiDARs. Applications include fruit detection and counting [44], up to plant phenotyping [71], health status assessment and growth monitoring [77]; [62]. While these methods were proved to be effective and accurate for detailed information extraction, they are often constrained to structured environments, such as greenhouses and specific acquisition conditions, such as controlled illumination or predefined positioning of the sensing devices, or they require the adoption of expensive high-resolution sensors [78], which limits their practical implementation. In order to address these issues, crop monitoring by agricultural ground robots has been proposed as a step forward to automated proximal measurement and characterization of high-value crops and soils [83] While much work has been done in the context of ground robots for harvesting and pick-

ing operations, the use of UGVs for in-field crop monitoring and assessment has been proposed more recently. UGVs can carry a number of sensing devices, thus potentially providing an efficient means to gather multi-modal information at a narrow scale. At the same time, they can be equipped with manipulators and actuators to perform targeted actions, such as selective spraying or fertilizing, with relatively high operating times.

Although UGVs offer enough payload to transport a number of bulky sensors, keeping low complexity and costs is a major requirement for infield implementation. In this respect, visual sensors mounted on ground robots have been shown to provide an efficient and affordable solution in a wide range of agricultural applications, including plant and fruit detection, fruit grading, ripeness detection, yield prediction, plant and fruit health protection and disease detection. In addition, visual sensors provide a rich source of information to support autonomous navigation functions such as localization, obstacle detection and situation awareness in general [112, 84].

Among visual sensors, portable consumer-grade RGB-D cameras, like Microsoft Kinect, have been receiving growing attention, as an effective means to recover in real-time 3D textured models of plants and extract plant and fruit features [21], although the application of this sensor remains mostly limited to indoor contexts. A novel family of highly portable, consumer depth cameras has been introduced by Intel in 2015 (R200 and D4xx, Santa Clara, CA, USA). These cameras are similar to the Kinect sensor in scope and cost, but use a different working principle based on IR stereo, which makes them more suitable for outdoor conditions. In addition, their output include RGB information, infrared images and 3D depth data, thus covering a wide range of information about the scene. The potential of these sensors for agricultural applications has been investigated in recent works [30, 24]. Following this research trend, this work explores the potential of a multi-view RGB-D system for geo-referenced image acquisition and mapping of a high-value crop, like a vineyard. The device is built following a modular approach and can be mounted on any agricultural vehicle to provide ground-based 3D reconstruction of the traversed crop rows. Data acquisition and processing can be carried out during vehicle operations, in a non-invasive and completely automatic way, while requiring low investment and maintenance costs.

One specific aspect addressed is accurate vehicle localization. Localization of the UGV is essential for correct merging of point cloud streams and thus for the construction of geo-referenced 3-D maps. In [60], the UGV pose estimation problem is formulated as a pose graph optimization to mitigate sensor drift and significantly improve state estimation accuracy using a Digital Elevation Model (DEM) and a Markov Random Field (MRF) assumption. Authors in [54] proposed a Simultaneous Localization And Mapping (SLAM) method for generating the map of an agricultural environment and simulated it on Gazebo and Robot Operating System (ROS) for the case of an apple farm, showing good results in fruit mapping. A well-established solution to the localization problem to fuse information from multiple sensors is Kalman filtering. In this work, we use the information form of the Kalman Filter as data fusion strategy for heterogeneous sensors. The reason is related to the high reliability of such algorithm, as confirmed by recent research (e.g., [3] and [135]). Other alternatives have been investigated in the

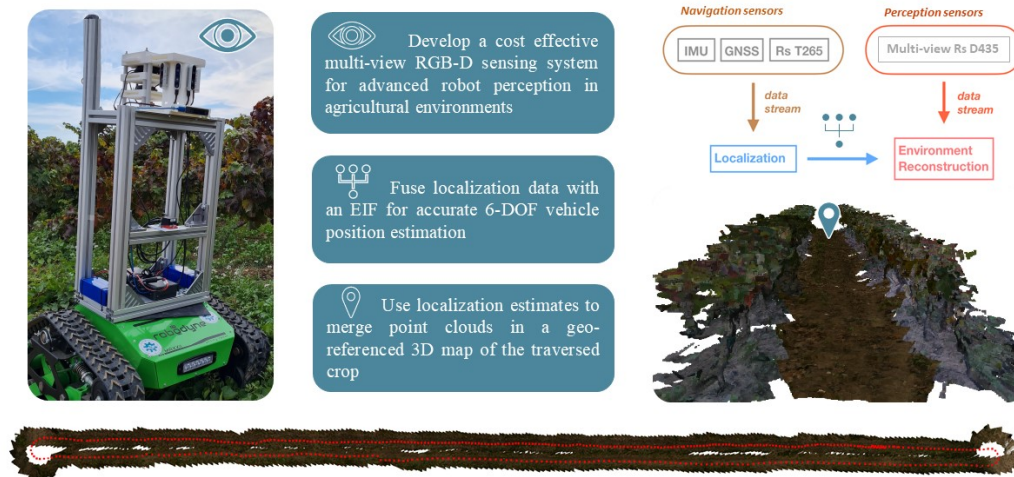
literature, including, for example, particle filtering, which however has proven to be less accurate for localization purposes [122] [108].

Section 4.2 presents the sensor suite used for data gathering and the algorithms to combine exteroceptive and proprioceptive measures. Section 5 proposes a novel approach for unsupervised system modelling based on the algorithm presented in section 4.2. Then tests the proposed Kalman Supervised Network on the simple case of a mass-spring-damper system and discusses the results obtained.

4.2 Multi-view Mapping

This section of the thesis is based on the published paper [127].

An RGB-D multi-view perspective for autonomous agricultural robots



Automated in-field data gathering is essential for crop monitoring and management and for precision farming treatments. To this end, consumer-grade digital cameras have been shown to offer a flexible and affordable sensing solution. This work describes the integration and development of a cost-effective multi-view RGB-D device for sensing and modelling of agricultural environments. The system features three RGB-D sensors, arranged to cover a horizontal field of view of about 130 deg in front of the vehicle, and a suite of localization sensors consisting of a tracking camera, an RTK-GPS sensor and an IMU device. The system is intended to be mounted on-board an agricultural vehicle to provide multi-channel information of the surveyed scene including color, infrared and depth images, which are then combined with localization data to build a multi-view 3D geo-referenced map of the traversed crop. The experimental demonstrator of the multi-sensor system is presented along with the steps for the integration of the different sensor

data into a unique multi-view map. Results of field experiments conducted in a commercial vineyard are included, as well, showing the effectiveness of the proposed system. The resulting map could be useful for precision agriculture applications, including crop health monitoring, and to support autonomous driving.

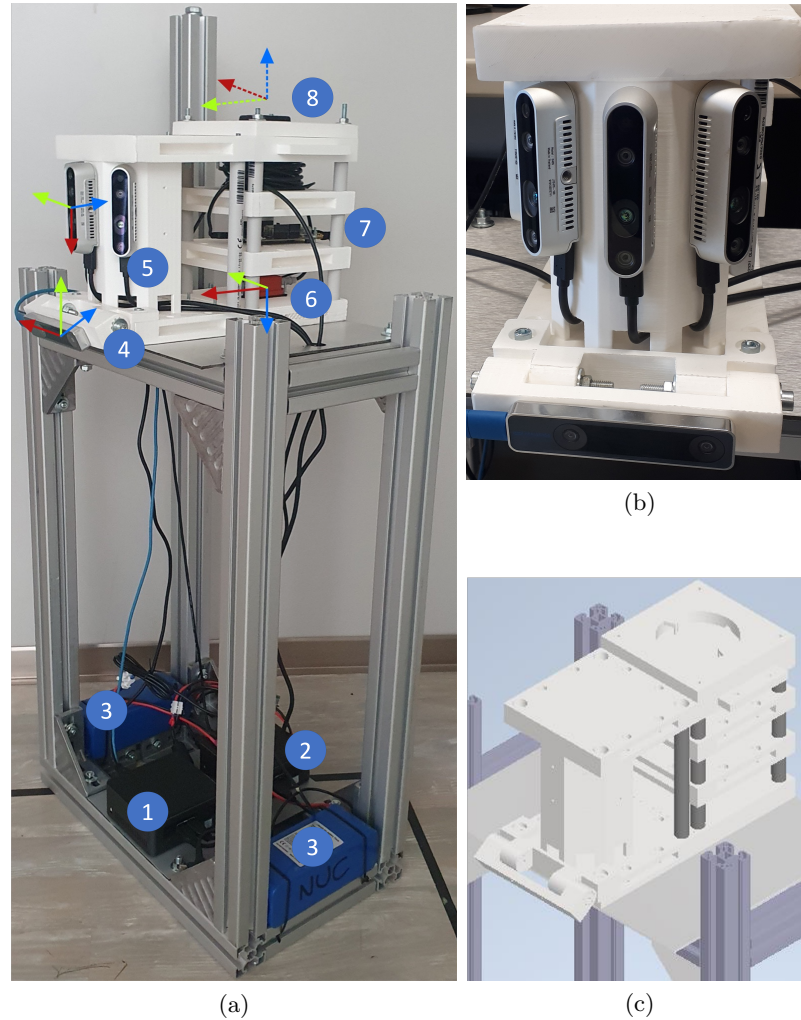


Figure 4.2.0.1: (a) Demonstrator of the UGV's sensor box: (1)-(2) Intel NUC Windows PCs; (3) Batteries, (4) T265 Camera, (5) D435 Cameras, (6) X-Sense MTI-300 IMU, (7) U-blox ZED F9P board, (8) Sensor Box GPS antenna. (b) Closeup of the multi-camera system. (c) CAD model of the sensor frame.

4.2.1 Multi-sensor Device

This section describes the development of a multi-sensor box for close range sensing and modelling of agricultural environments. The sensor suite is intended to be mounted on board an agricultural robot and is designed to be self-contained, both from a computa-

tional and energy point of view, and independent of the particular vehicle architecture.

Hardware Design The sensor suite is shown in Figure 4.2.0.1 (a). It consists of two sensor arrays, namely a *Perception Sensors* array and a *Navigation Sensors* array. The perception sensors include three Intel RealSense D435 RGB-D cameras arranged to cover a wide horizontal field of view of about 130 deg in front of the vehicle, which extends up to about 145 deg when considering infrared depth information only. The mounting case allows one to alternatively place up to two cameras in lateral configuration, e.g., to keep the image plane parallel to a crop row for tasks such as row following and/or monitoring. A closeup of the multi-camera system is shown in Figure 4.2.0.1 (b). The navigation sensors comprise one Intel RealSense tracking camera T265, one X-Sense IMU MTi-300 and two U-Blox GPS Zed-F9P providing RTK-GPS data in rover-base configuration. All sensors are integrated in a 3-D printed PLA box (see Figure 4.2.0.1 (c)), which was designed following a modular approach, so that it can be assembled in multiple ways according to the specific needs of the test field. The described sensor suite can be fixed to the vehicle through a metal frame, built with aluminum bars and plates and designed to be stable and of adjustable height. Two Intel NUC7i7DNHE computers are used for data gathering. The PCs, powered by lithium batteries, are fixed at the bottom of the metal frame. Overall, the proposed sensor box provides a flexible and self-contained data gathering device with a cost of about 6.5k €(i.e., 27% for the two processing units, 45% for the IMU, 14% for the cameras, 7% for the GPS, and 7% for the batteries).

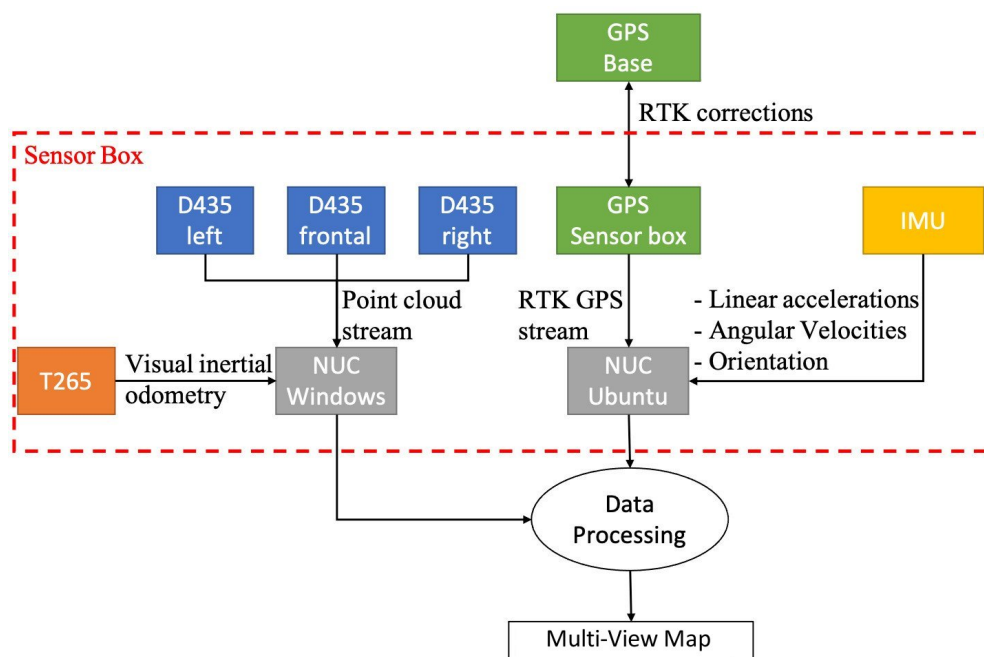


Figure 4.2.1.1: Sensor box data flow.

Acquisition Software Design The data gathering pipeline of the sensor suite is shown in Figure 4.2.1.1. The sensor box provides two processing units, one running Ubuntu and the other running Windows 11. The NUC Ubuntu is devoted to gathering positional measurements produced by the GPS sensor and the IMU sensor. Data acquisition is made through ROS drivers using a dedicated ROS node for each sensor. Then, all the acquired data are stored in ROS bags.

For image acquisition and storage, a software package, named SensorBox, was developed using the Intel RealSense SDK 2.0 (v. 2.49), running on NUC Windows. The software architecture of the whole package is divided into two executables: *MultiBagReader* and *MultiBagWriter*. The scheme of the first executable (*MultiBagWriter*) is shown in Figure 4.2.1.2. It works in a producer-consumer logic, where the Intel RealSense cameras connected to the processing unit are first opened to produce the data which is then consumed, i.e. displayed, to show the acquired field of view and/or the computed visual odometry. Then, when the user starts the acquisition, the data are encapsulated in several ROS bags, stored on the local hard disk of the NUC. In this way, each camera produces a bag file at the maximum achievable rate (up to 30 fps), without any further processing to prevent frame drops. It is worth noticing that each camera works in a free run mode and, thus, their frames are not temporally synchronized, i.e. acquired exactly at the same time instant. The second executable (*MultiBagReader*) opens the bags, divides the RealSense pipelines to have single streams in each pipeline, and then reports all the acquired frames to a global temporal reference, thus performing software synchronization. The software features a user interface for both writing and reading modules, as shown in Figure 4.2.1.3. The open-source code of the software is available on STIIMA GitHub.

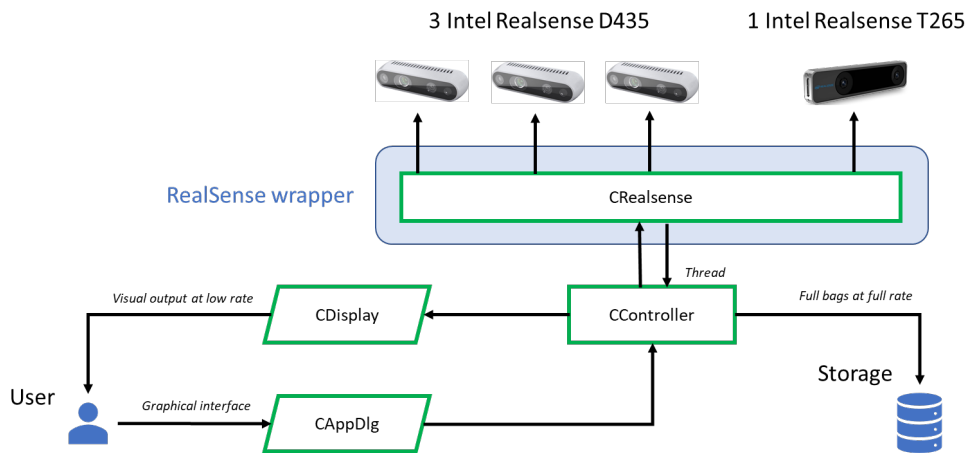
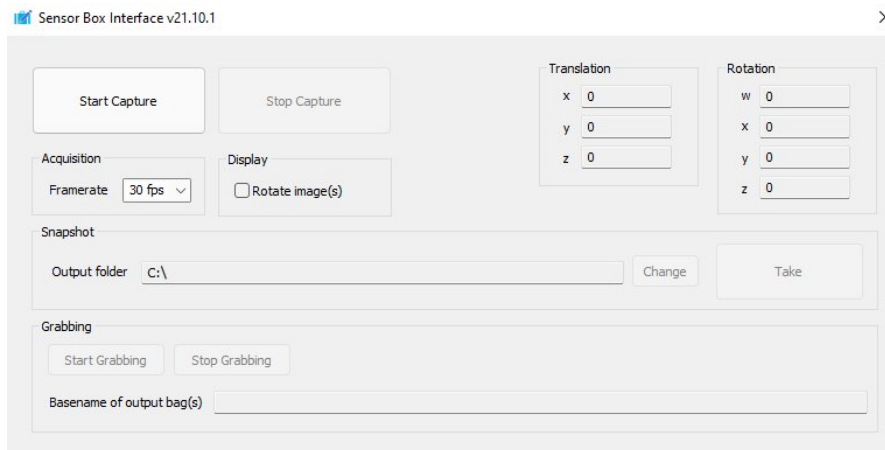
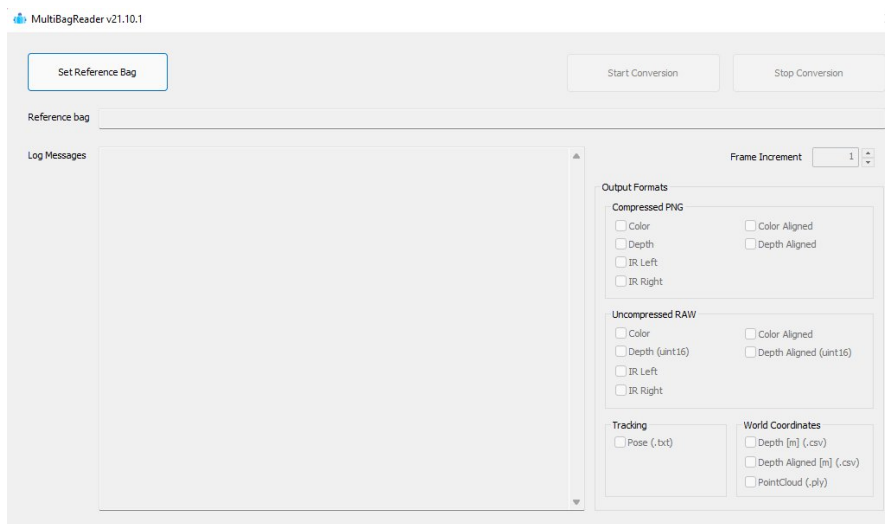


Figure 4.2.1.2: Schematic of the image acquisition software.

Sensor Synchronization and Calibration The association of heterogeneous data requires temporal and spatial calibration. For time synchronization, a timestamp-based



(a)



(b)

Figure 4.2.1.3: User interface of the multi-view camera system: (a) interface for data acquisition and storage (MultiBagWriter); (b) interface for reading stored image databases (MultiBagReader).

approach was adopted, whereby each sensor observation was marked with a timestamp. In addition, to register all sensor data with respect to a common reference frame, spatial calibration was performed to estimate the relative position and orientation of the sensors with respect to each other. Spatial calibration was performed by construction, considering that all the sensors are located in the sensor box at fixed positions. This proved to be sufficiently accurate for the purpose of this work, although optimization strategies, such as the one proposed by the authors in [101], can be also adopted to further improve

the registration accuracy.

4.2.2 Multi-view 3D Mapping

The data acquired by the sensor suite are processed to build a multi-view map of the traversed environment, following multiple stages. First, data from GPS, IMU and T265 sensors are fused by an EIF to generate pose estimates. Successively, the point clouds obtained by each of the three RGB-D cameras are assembled into a unique map, using the EIF pose estimates and the known relative poses between the sensors. The map can be then converted into a 3D mesh representation for efficient storage and inspection, as well as, for import in a dedicated simulation environment, as will be described later in Section 5. In more detail, with reference to Figure 4.2.0.1, let us introduce the reference frames denoted with the following subscripts:

- *sb*: Sensor Box frame (Figure 4.2.0.1, a-6)
- *w*: East-North-Up world frame (Figure 4.2.0.1, a-8)
- *cam*: Camera Frame (Figure 4.2.0.1, a-5)
- *s*: Reference frame for the *s*-th sensor (RTK-GPS, T265, IMU).

Furthermore, quantities of interest are presented here to facilitate the understanding of the mapping reconstruction process:

- $p_{B-A}(t) \in \mathbb{R}^{1,3}$: position vector of reference frame *A* at time *t* expressed in reference frame *B*
- $\mathbf{q}_{B-A}(t) \in \mathbb{C}^{1,4}$: quaternion orientation of reference frame *A* at time *t* expressed in reference frame *B*. Note that only quaternions are denoted in bold.

Quaternion Analysis Quaternion representations are convenient for composition of rotations and coordinate transformations. Here we provide the reader with an overview of basic concepts of quaternion analysis, the interested reader is referred to the literature (e.g., [22]) for more details. The unit quaternion $\mathbf{q} = [q_x, q_y, q_z, q_w]$, is uniquely mapped to a rotation matrix *R* and describes the transformation between two reference frames as a rotation of a certain angle θ around the direction vector \vec{n} following Equation 4.2.2.1.

$$\mathbf{q} = [n_x \sin \frac{\theta}{2}, n_y \sin \frac{\theta}{2}, n_z \sin \frac{\theta}{2}, \cos \frac{\theta}{2}] \quad (4.2.2.1)$$

Denoting with s_1 and \vec{v}_1 respectively the scalar and vector part of quaternion $\mathbf{q}_1 = [\vec{v}_1, s_1] = [v_{1x}, v_{1y}, v_{1z}, s_1]$, and with s_2, \vec{v}_2 the scalar and vector part of quaternion \mathbf{q}_2 , the operation of quaternion product can be expressed as

$$\mathbf{q}_2 \mathbf{q}_1 = [s_2 \vec{v}_1 + s_1 \vec{v}_2 + \vec{v}_2 \times \vec{v}_1, s_2 s_1 - \vec{v}_2 \cdot \vec{v}_1] \quad (4.2.2.2)$$

where quaternion product symbol has been omitted for readability, whereas $\vec{v}_2 \cdot \vec{v}_1$ denotes dot product between vectors \vec{v}_1 and \vec{v}_2 and finally $\vec{v}_2 \times \vec{v}_1$ denotes cross product between the two vectors. Quaternion product is a non-commutative operation and returns a quaternion that represents the orientation obtained after the sequence of transformations \mathbf{q}_1 and then \mathbf{q}_2 . The norm of a quaternion \mathbf{q} is denoted as $\|\mathbf{q}\|^2 = q_x^2 + q_y^2 + q_z^2 + q_w^2$. The conjugate of quaternion $\mathbf{q} = [\vec{v}, s]$ is represented as $\mathbf{q}^* = [-\vec{v}, s]$, while its inverse $\mathbf{q}^{-1} = \frac{\mathbf{q}^*}{\sqrt{\|\mathbf{q}\|^2}}$. For a unit quaternion we have $\|\mathbf{q}\|^2 = 1$ so its conjugate coincides with its inverse. All quaternions describing orientation in 3-D space are unit quaternions. The normalized quaternion denoted as $\|\mathbf{q}\| = \frac{\mathbf{q}}{\sqrt{\|\mathbf{q}\|^2}}$ has a unit norm and each of its

components are divided by $\sqrt{\|\mathbf{q}\|^2}$. Let us denote with \mathbf{q}_{B-A} the quaternion describing orientation of frame A with respect to frame B written in frame B , its inverse is $\mathbf{q}_{B-A}^{-1} = \mathbf{q}_{A-B}$. Then, composition of rotations can be obtained in a convenient form as

$$\mathbf{q}_{C-A} = \mathbf{q}_{C-B}\mathbf{q}_{B-A} \quad (4.2.2.3)$$

Consider the position vector \vec{p}_A in frame A , then its projection in reference frame B can be obtained as

$$[\vec{p}_B, 0] = \mathbf{q}_{B-A}[\vec{p}_A, 0]\mathbf{q}_{A-B} \quad (4.2.2.4)$$

Finally, denoting with $\vec{\omega}_B(t)$ the angular velocity of moving frame B in its reference frame A , the derivative of the quaternion $\mathbf{q}_{A-B}(t)$ expressed in the inertial frame A can be computed as

$$\frac{d\mathbf{q}_{A-B}(t)}{dt} = \frac{1}{2}\mathbf{q}_{A-B}(t)[\vec{\omega}_B(t), 0] \quad (4.2.2.5)$$

Mapping Algorithm The mapping algorithm proceeds according to the following three steps:

1. *Pose estimation*: position and orientation of the sensor box can be estimated from different sensors (RTK-GPS, T265, IMU) and fused within an EIF. In general, the sensor s can provide information about its position and orientation expressed either in the world frame or with respect to its initial pose. In the former case, Equations (4.2.2.6) give the sensor box orientation \mathbf{q}_{w-sb} and position p_{w-sb} in the world frame, knowing \mathbf{q}_{s-sb} and p_{sb-s}

$$\begin{aligned} \mathbf{q}_{w-sb}(t) &= \mathbf{q}_{w-s}(t)\mathbf{q}_{s-sb} \\ [p_{w-sb}(t), 0] &= [p_{w-s}(t), 0] - \mathbf{q}_{w-sb}(t)[p_{sb-s}, 0] \end{aligned} \quad (4.2.2.6)$$

If the sensor provides $p_{s_0-s}(t)$ and $\mathbf{q}_{s_0-s}(t)$ with respect to its initial condition s_0 , Equations (4.2.2.7) compute p_{w-sb} and \mathbf{q}_{w-sb}

$$\begin{aligned} \mathbf{q}_{w-sb}(t) &= \mathbf{q}_{w-sb_0}\mathbf{q}_{sb_0-s}(t)\mathbf{q}_{s-sb} \\ [p_{w-sb}(t), 0] &= \mathbf{q}_{w-sb_0}\mathbf{q}_{sb-s}[p_{s_0-s}(t), 0]\mathbf{q}_{s-sb}\mathbf{q}_{sb_0-w}^- \\ &\quad + \mathbf{q}_{w-sb}(t)[p_{sb-s}, 0] + \\ &\quad + [p_{w-sb_0}, 0] \end{aligned} \quad (4.2.2.7)$$

Apart from $\mathbf{q}_{sb-s} = \mathbf{q}_{s-sb}^{-1}$, initial position and orientation of the sensor box in the world frame (p_{w-sb_0} and \mathbf{q}_{w-sb_0}) are needed to compute p_{w-sb} and \mathbf{q}_{w-sb} . In Equations (4.2.2.7) notation $[p, 0]$ indicates the quaternion with zero real part corresponding to position vector p .

Please note that in both Equations (4.2.2.6) and (4.2.2.7) quaternion multiplication is omitted for ease of notation.

The predictive model used to implement the EIF is expressed by Equations (4.2.2.8)

$$\begin{aligned}\tilde{p}_{t+1|t} &= \tilde{p}_{t|t} + \tilde{v}_{t|t} \cdot \Delta t \\ [\tilde{v}_{t+1|t}, 0] &= [\tilde{v}_{t|t}, 0] + (\tilde{\mathbf{q}}_{t|t}[a_{sb}, 0]\tilde{\mathbf{q}}_{t|t}^{-1} - [g, 0]) \cdot \Delta t \\ \tilde{\mathbf{q}}_{t+1|t} &= \|(\tilde{\mathbf{q}}_{t|t} + \frac{\Delta t}{2} \cdot \tilde{\mathbf{q}}_{t|t}[\omega_{sb}, 0])\|\end{aligned}\quad (4.2.2.8)$$

where position \tilde{p} , velocity \tilde{v} , and quaternion $\tilde{\mathbf{q}}$ describe the state of the sensor box over time expressed in the w frame. Subscript $t+1|t$ denotes predicted value and $t|t$ indicates posterior value corrected with measurements. Expression $\|\mathbf{q}\|$ represents quaternion normalization. Angular velocity ω_{sb} and linear acceleration a_{sb} are measurements provided by the IMU and expressed in the sb frame, and g denotes gravity acceleration in the world frame.

2. *Point cloud assembly* The pose estimates are then used to transform and assemble the point clouds provided by each RGB-D camera in its own frame. Denoting with $P_{cam} \in \mathbb{R}^{3,1}$ a generic point of the point cloud in the camera frame cam and with $T_{cam} \in \mathbb{R}^{4,4}$ the homogeneous transformation matrix from frame cam to frame sb , Equation (4.2.2.9) expresses P_{cam} in the sb frame as P_{sb} .

$$[P_{sb}^T, 1]^T = T_{cam}[P_{cam}^T, 1]^T \quad (4.2.2.9)$$

The homogeneous transformation matrix T_{sb} from the sensor box frame to the inertial frame can be obtained by Equation (4.2.2.10) using the pose estimated by the EIF

$$T_{sb} = \begin{bmatrix} \text{rotm}(\tilde{\mathbf{q}}_{t|t}) & \tilde{p}_{t|t}^T \\ [0, 0, 0] & 1 \end{bmatrix} \quad (4.2.2.10)$$

where $\text{rotm}(\mathbf{q}) \in \mathbb{R}^{3,3}$ returns the rotation matrix uniquely assigned to quaternion \mathbf{q} . Finally, 3D points can be transformed from the sb to the w frame using Equation (4.2.2.11)

$$[P_w^T, 1]^T = T_{sb}[P_{sb}^T, 1]^T \quad (4.2.2.11)$$

3. *Map merging.* Assembled point clouds gathered by the three cameras and transformed in the world frame are then merged together and down sampled (voxel grid size= 0.01 m).

4.2.3 Simulation Environment

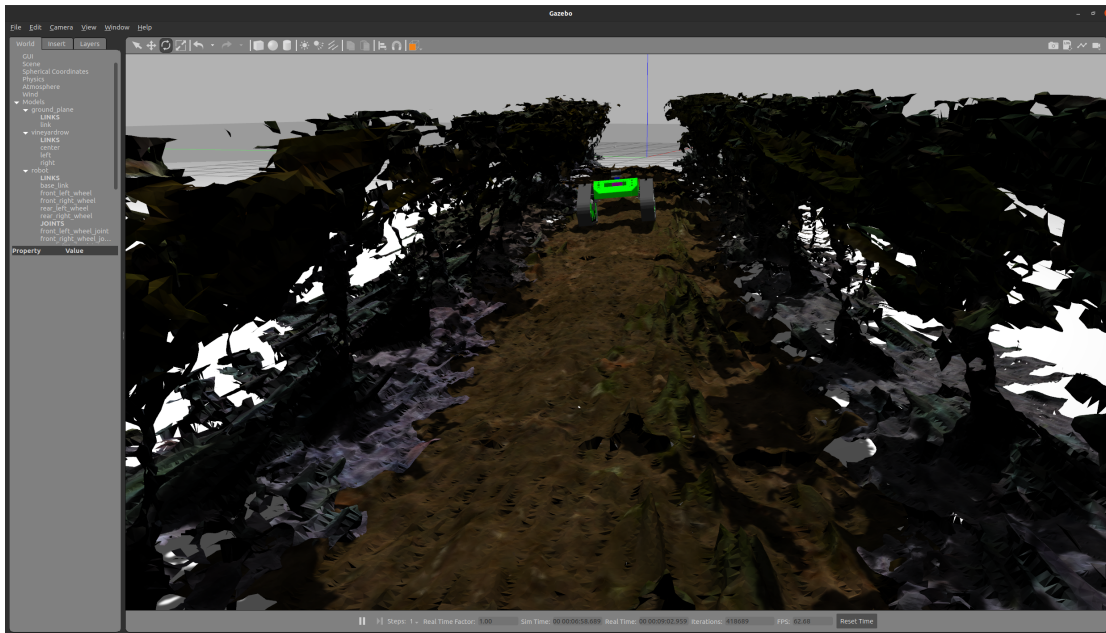


Figure 4.2.3.1: Gazebo simulation environment composed of a vineyard row and a mobile robot.

The possibility to simulate robots in outdoor environments under different conditions is of paramount importance in agricultural robotics. In this regard, the most famous simulator framework used by the robotic community is Gazebo. Gazebo is an open-source simulator born in 2002 for robotics. After over 15 years of development, Gazebo is undergoing a significant upgrade and modernization. Today, Gazebo is part of Ignition, which is a collection of open-source software libraries designed by Open Robotics to simplify the development of high-performance applications. Creating Gazebo models means creating SDF (Simulation Description Format) files to define SDF Model Objects. The primary element composing an SDF file is the link. A link contains the physical properties of one body of the model. Each link may have many collisions and visual elements. Usually, these two elements define the 3D mesh file describing the 3D surface of the object. Gazebo requires that mesh files be formatted as STL, Collada, or OBJ, with Collada and OBJ preferred formats.

In this work, the point cloud map, obtained as described Section 4.2.2, is modeled using a meshing algorithm, which allows one to generate a mesh-based representation for map import in the Gazebo simulation environment, according to the following steps:

- **Downsampling:** to down sample the obtained dense map is a not mandatory task that enables to speed up the mesh reconstruction process and improves the outcome of the whole process. To accomplish this task, the samples are generated

according to a Poisson-disk distribution [27];

- **Normals computation:** the knowledge of the normals is necessary to reconstruct the surface of the elements composing the map. Normals are computed on the basis of the 10 closest points;
- **Surface reconstruction:** this step regards the reconstruction of the surface starting from the set of points and normals. In this case, the Ball Pivoting algorithm is used [15] to compute a triangle mesh. It is based on the principle that three points form a triangle if a ball of a user-specified radius touches them without containing any other point;
- **Texture mapping:** the texture mapping is build with triangle by triangle parametrization;
- **Color transfer:** this step concerns the process of projecting a 2D image to a 3D model's surface for texture mapping, the so called *UV mapping*. Once a UV map is available, the color can be transferred to the reconstructed surface;
- **Save the mesh:** finally, the mesh is ready to be exported in a suitable format.

Thus, thanks to the created mesh files, it is possible to develop a Gazebo SDF model object describing the reconstructed vineyard row. As an example, Figure 4.2.3.1 showcases a mobile robot crossing a vineyard row developed by following the procedure described above.

4.2.4 In-field Testing

The multi-sensor system is mounted and integrated the polibot, and it is tested in field conditions, as shown in Figure 1.4.0.2 (a). Dedicated tests are performed in a commercial vineyard in San Donaci, Apulia region, Italy. Specifically, the robot is guided to follow closed-loop trajectories around different crop rows while gathering the sensor data. The data were then processed offline to recover the 6DoF path and the 3D map of the environment.

In this section, first, the localization performance of the proposed system is analyzed in terms of accuracy and repeatability. Then, the mapping results are discussed.

4.2.4.1 Localization performance

Four closed-loop runs are considered, performed along two different crop rows of about 120 m length, referred to as Test 1 to Test 4 in the following. They belong to two field campaigns carried out in September and October 2021, respectively, during different times of the day. Three localization sources are compared, namely RTK-GPS only, T265 only and EIF. A projection in Google Earth view of the trajectories reconstructed by EIF for the four paths is shown in Figure 4.2.4.1. Numerical results for all runs are collected in Table 4.2.4.1, showing the discrepancy in the East-North-Up (w) frame between the

starting and ending points of the trajectory, expressed in terms of 3D Euclidean distance (D), 2D Euclidean distance in the motion plane (D_{EN}) and altitude distance (D_U), and the standard deviation of altitude measurements (σ_U) along the entire path.

Table 4.2.4.1: Comparison of different localization sources along four robot paths (Test 1 to 4): discrepancy in the East-North-Up frame between the starting and ending points of the trajectory expressed in terms of 3D Euclidean distance (D), 2D Euclidean distance in the motion plane (D_{EN}), altitude distance (D_U), and standard deviation of altitude measurements (σ_U) along the entire path.

<i>Test</i>	<i>Source</i>	$D[m]$	$D_{EN}[m]$	$D_U[m]$	$\sigma_U[m]$
1	RTK-GPS	1.501	1.511	0.026	0.451
	T265	1.276	1.275	0.046	2.425
	EIF	1.514	1.513	0.032	0.228
2	RTK-GPS	0.579	0.579	0.013	0.452
	T265	5.909	4.921	3.270	2.199
	EIF	0.579	0.579	0.011	0.209
3	RTK-GPS	1.852	1.517	1.063	0.370
	T265	7.264	5.811	4.359	1.440
	EIF	1.833	1.484	1.077	0.355
4	RTK-GPS	1.290	1.073	0.717	0.670
	T265	5.341	4.760	2.424	1.497
	EIF	0.739	0.356	0.647	0.428

The robot path as estimated by each localization source is reported in Figure 4.2.4.2 for Test 1. In this test, pose estimates using only RTK-GPS (Figure 4.2.4.2 (a)) are consistent as long as RTK correction is available. The starting and ending points are close to each other and the altitude estimate is stable except when the connection to the base GPS is lost and the RTK correction is missing (red diamonds in Figure 4.2.4.2 (a)). Figure 4.2.4.2 (b) shows the path as reconstructed by the T265 proprietary visual-inertial SLAM algorithm in its own frame and successively transformed in the world frame.

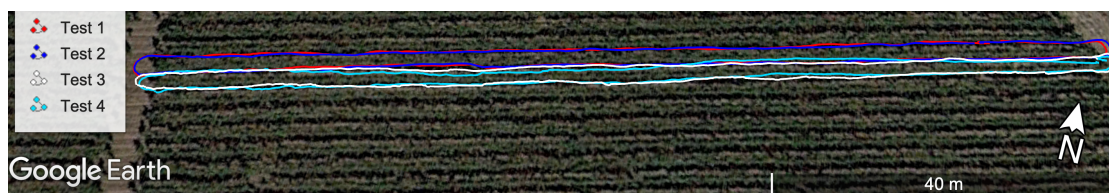


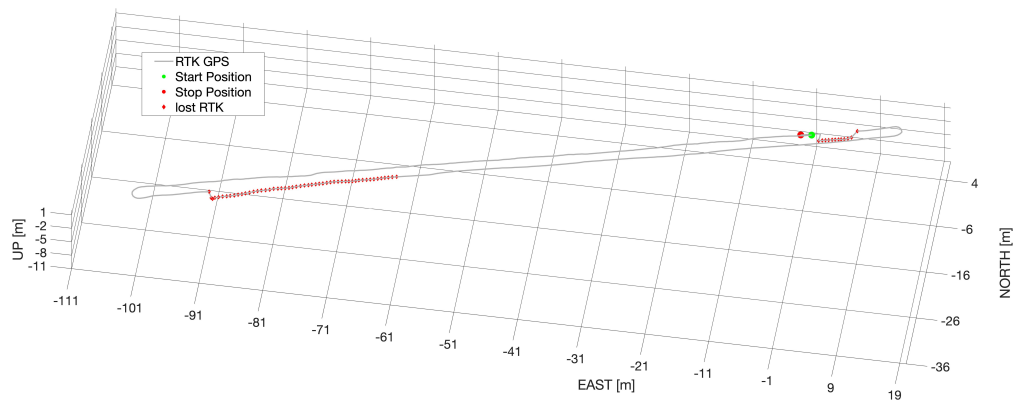
Figure 4.2.4.1: Google Earth view of the four paths estimated by EIF in a commercial vineyard, San Donaci, Apulia Region, Italy ($40^{\circ}27'16.2''$ N $17^{\circ}54'30.6''$ E).

Compared to RTK-GPS path, the distance between starting and ending points esti-

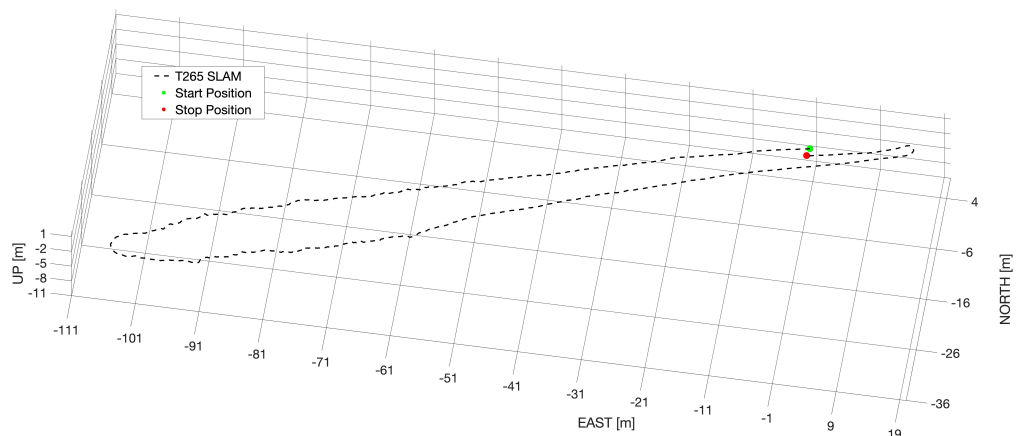
mated by the T265 camera is 15.6% smaller in terms of East-North coordinates (D_{EN}) but 77% larger in terms of altitude (D_U). Low accuracy of vertical displacement estimates leads to large standard deviation of altitude measurements (σ_U), which for T265 is of 2.43 m, about 5 times larger than the one obtained by RTK-GPS. Figure 4.2.4.2(c) shows the path reconstructed by the EIF. The EIF uses linear acceleration and angular velocity measures to make state predictions using the model described by equations (4.2.2.8), and corrects its predictions using measures of RTK-GPS (world position and velocity), IMU (world orientation) and T265 (relative position and orientation). The EIF estimates a difference between starting and ending points 0.18% larger than the RTK-GPS in terms of East-North position and 23% larger in terms of vertical displacement. However, the standard deviation of altitude measures is 0.22 m for the EIF, i.e., 49% smaller than the one provided by RTK-GPS, suggesting an overall improvement in position estimate when fusing measurements with the EIF. This improvement is due to the fact that the RTK corrections are not available when connection is lost with the base GPS, whereas the EIF adjusts the position estimates for these instants using predictions with IMU data and short-term measures of the T265 camera when the covariance of position and velocity provided by the GPS grows.

When considering a second run along the same row (Test 2), similar results are obtained (see Figure 4.2.4.3 (a) and the corresponding row in Table 4.2.4.1) in terms of σ_U which attests to 0.45 m for RTK-GPS, 2.20 m for T265 camera and 0.21 m for EIF. The discrepancy between starting and ending points (D) for T265 is higher than the one obtained from RTK-GPS indicating that visual inertial odometry should be only used for short-term displacement estimation. Again, the use of EIF allows for a reduction of σ_U of 53% with respect to GPS and of 90% with respect to T265, while preserving loop closure accuracy.

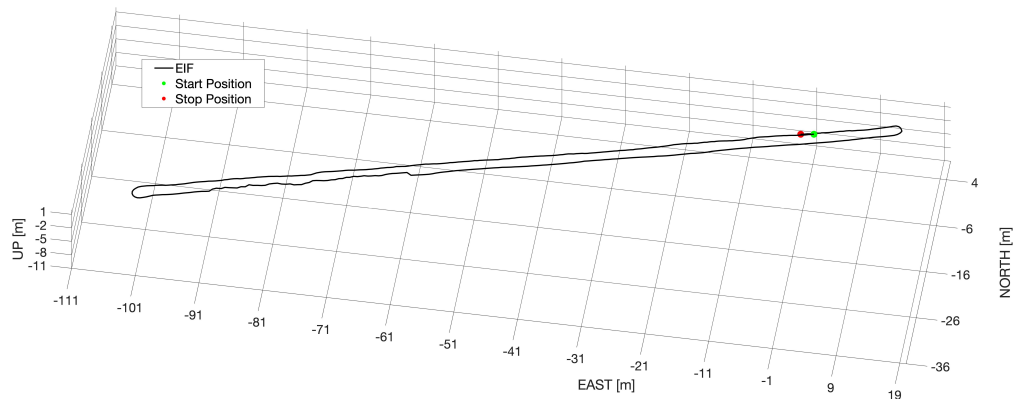
The localization results for a path along a different crop row (Test 3) is reported in Figure 4.2.4.3 (b). In this case, the T265 results in a substantially degraded estimation, and EIF mainly relies on GPS leading to σ_U of 0.35 m. On the contrary, Figure 4.2.4.3 (c) refers to an example where the quality of GPS signal is poor in several parts of the trajectory (Test 4). Again, EIF is able to compensate for the GPS outages mainly relying on T265 information showing better performance for all the metrics.



(a)

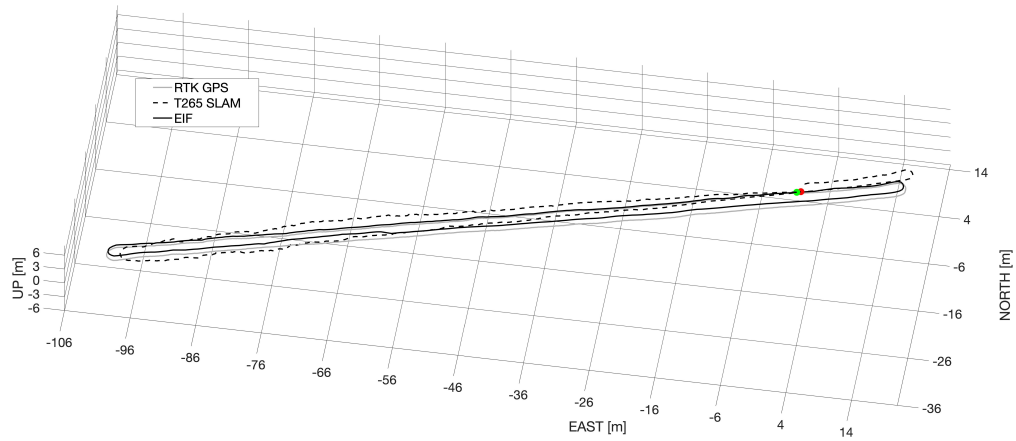


(b)

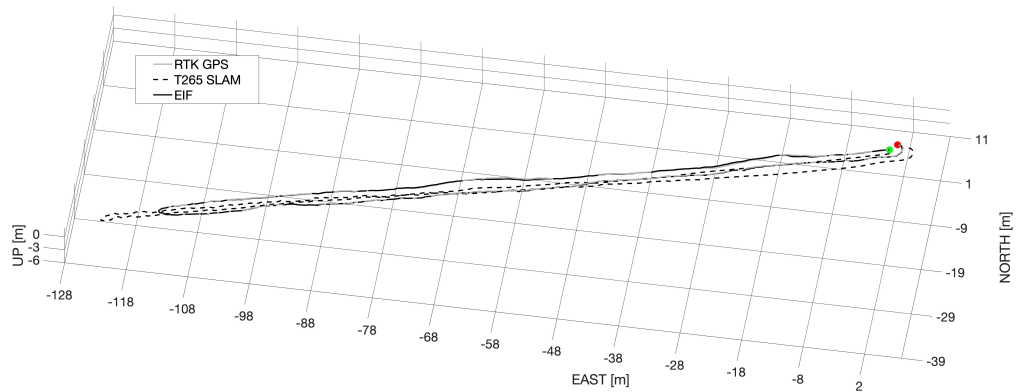


(c)

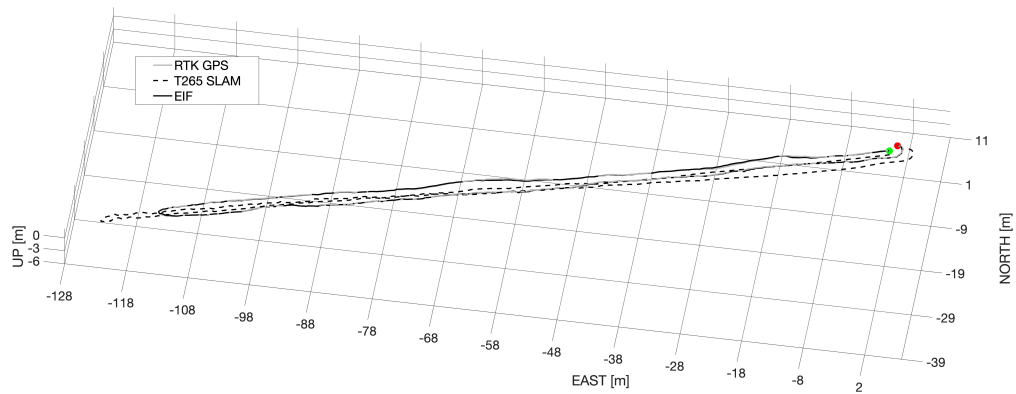
Figure 4.2.4.2: Localization results for Test 1: (a) from RTK-GPS only; (b) from T265 camera only; (c) after EIF fusion of RTK-GPS, IMU and T265 measurements. In (a), red diamonds are overlaid in two different zones without RTK coverage due to connection loss.



(a)



(b)



(c)

Figure 4.2.4.3: Localization results (a) for a second run along the same path of Test 2 (b) for Test 3 (c) for Test 4; RTK-GPS only (solid grey line), T265 only (dashed black line) and EIF (solid black line). Start and stop positions for EIF trajectory are denoted by green and red dot, respectively.

4.2.4.2 Mapping

For each geo-referenced position, the corresponding multi-view data can be recovered. As an example, Figure 4.2.4.4 shows the robot path (Test 1) overlaid over Google Earth view with three pinpointed positions, whereas the corresponding multi-view output is displayed in Figure 4.2.4.5.



Figure 4.2.4.4: EIF-derived path overlaid on Google Earth view for Test 1. Three successive positions of the robot are pinpointed. For these positions, the corresponding visual data are shown in Figure 4.2.4.5.

Point clouds are collected and assembled in the w frame using estimates of both absolute position and orientation of the sensor box. In Figure 4.2.4.6, the EIF observer output is used to merge point clouds collected by the frontal camera. Figure 4.2.4.8(a) shows, instead, about 20 m of merged point clouds from all cameras using 6DoF odometry provided by the EIF.

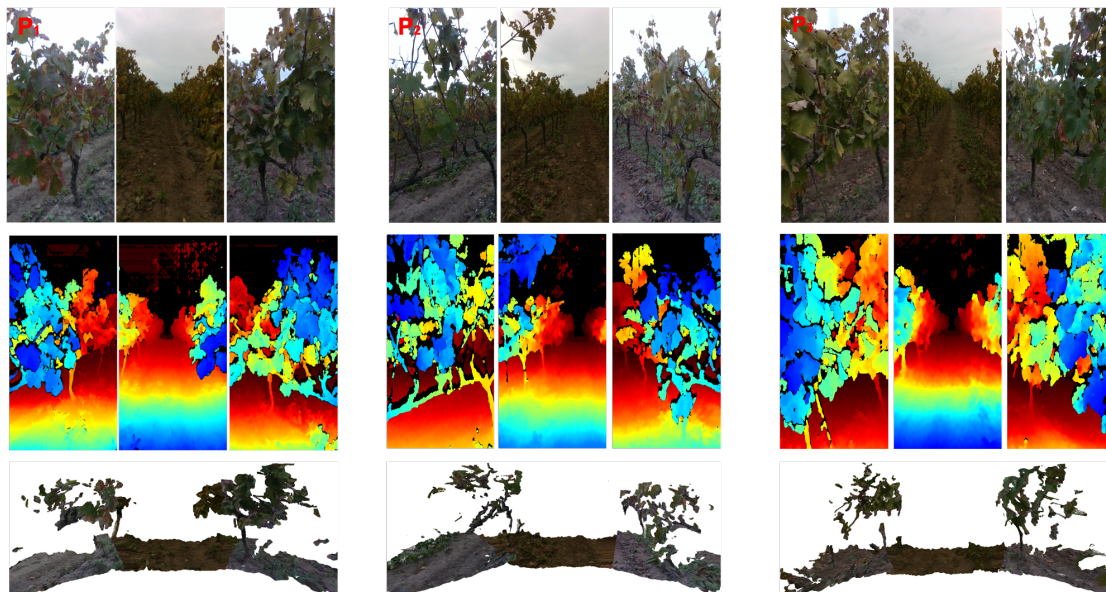


Figure 4.2.4.5: Output of the multi-view camera system for three robot locations along the path (Test 1): (first row) color images, (second row) depth images obtained from IR stereo reconstruction and (third row) multi-view 3D point cloud.

This map can be processed to extract high-level information about the crop, such as vegetation indexes and morphological information. As an example, Figures 4.2.4.8(b) and (c) show the map of Figure 4.2.4.8 (a) augmented with Green-Red Vegetation Index (GRVI) and crop elevation information, respectively.

Accurate localization information is essential to assemble subsequent point clouds acquired by the D435 cameras and build the environment map. This can be clearly seen in Figure 4.2.4.7, where two different 6DoF localization sources are compared. In detail, Figure 4.2.4.7(a) shows a group of point clouds badly assembled with synced data of GPS for position and IMU for orientation when RTK correction are missing. Figure 4.2.4.7(b) is obtained using the EIF for the same time span, clearly showing the improvement in point cloud assembling.



Figure 4.2.4.6: Mapping results (Test 1): upper view of the terrain map reconstructed by the central camera. The robot trajectory estimated by the sensor fusion approach is also overlaid.

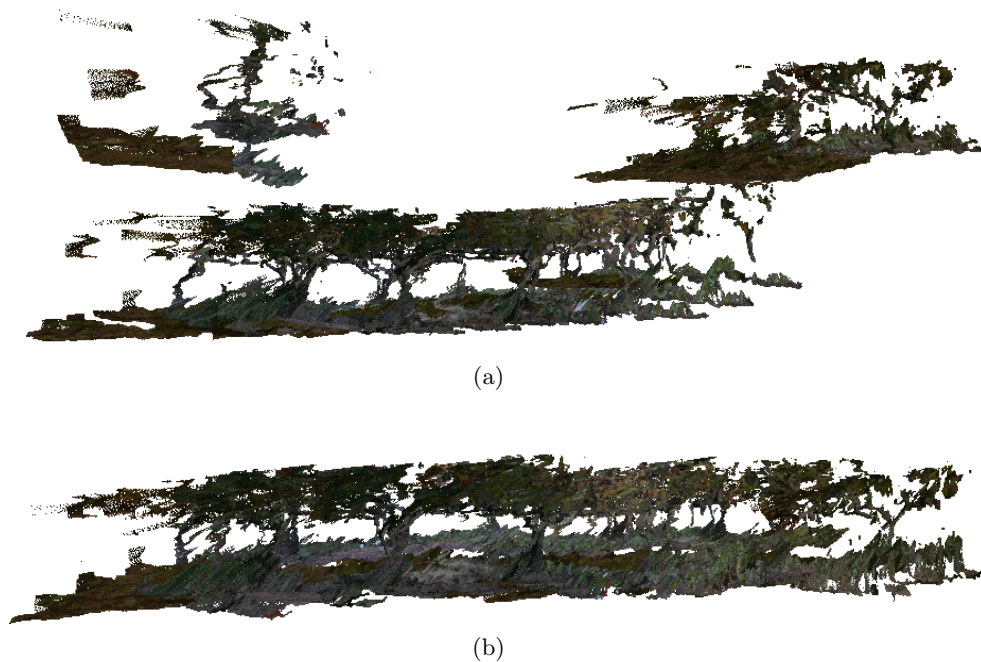


Figure 4.2.4.7: Mapping results (Test 1): closeup of loop closure before (a) and after (b) EIF correction.

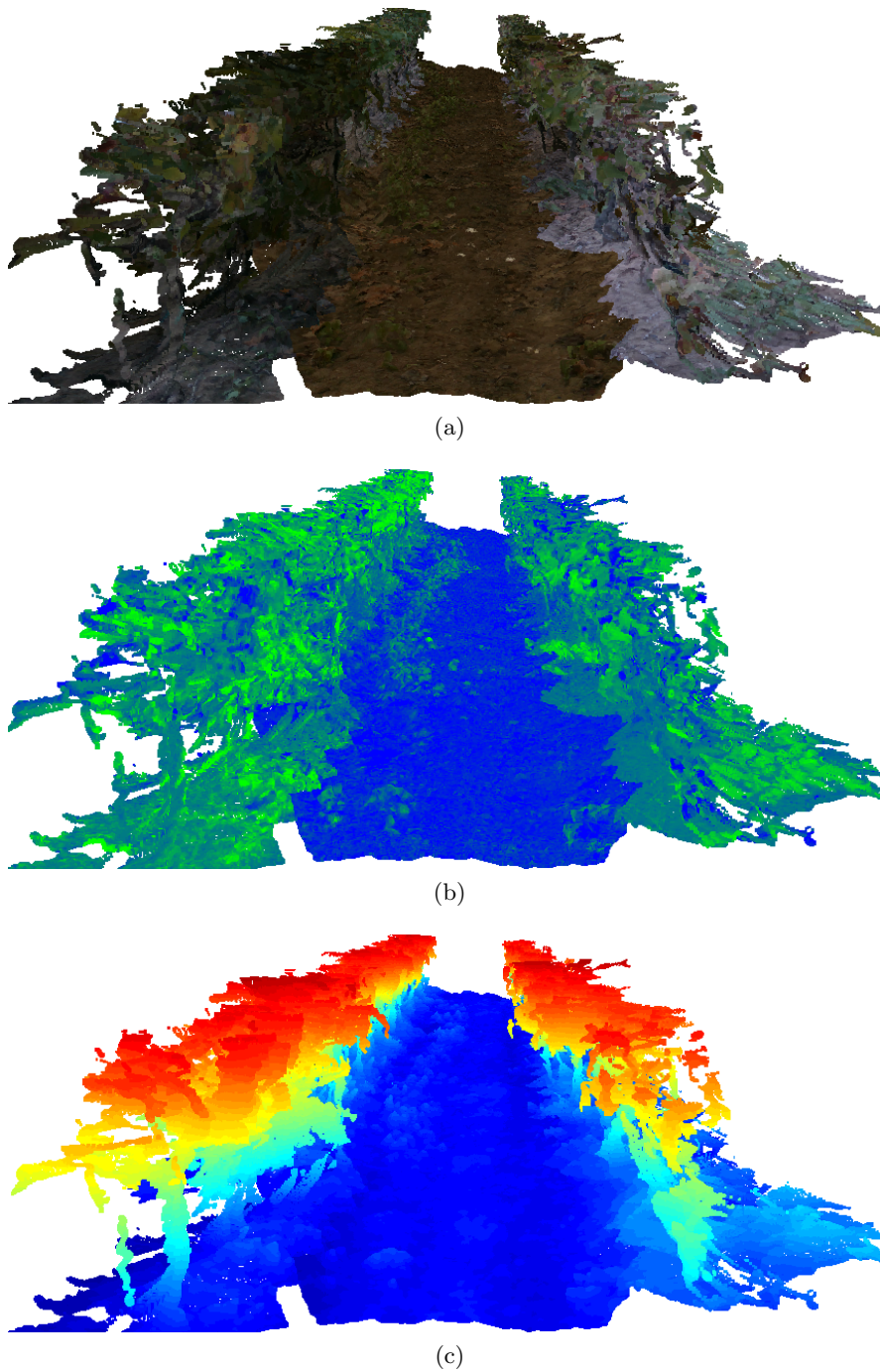


Figure 4.2.4.8: Closeup of the multi-view map for Test 1 (first 20 m): (a) RGB, (b) GRVI and (c) elevation map. In (b), green points refer to vegetation, whereas blue points correspond to non-vegetated parts. Lighter green denotes higher GRVI values. In (c), a jet colormap is used to represent point height with respect to ground.

4.2.4.3 Considerations

In this section, the development, implementation and testing of a multi-view RGB-D sensing device is presented. The system is intended to be mounted on an agricultural ground robot for in-field proximal monitoring of high-value crops. A multi-view mapping approach to combine information from multiple visual and localization sensors and produce a high-resolution 3D reconstruction of agricultural environments is described. It is based on an EIF algorithm to fuse information from RTK-GPS, IMU and visual-inertial SLAM to obtain an accurate estimation of the vehicle position in the field. Then, on the basis of localization data, subsequent point clouds reconstructed by the RGB-D sensors during robot motion can be assembled to generate a high-resolution map of the surveyed environment. Results of dedicated tests performed in a commercial vineyard are presented, showing the effectiveness of the proposed system for in-field data gathering in an automatic and non-invasive way.

Future work will include the processing of the maps using supervised or unsupervised classification methods to generate semantic representations of the environment, which can be used to improve vehicle autonomy and safety. Research will focus on the integration of output maps into Farm Management Information Systems (FMIS) to enable map-based control of agricultural applications and machinery. In this respect, future efforts will be devoted to address the real-time challenge by using the multi-view maps for online navigation of autonomous agricultural vehicles. Furthermore, methods for identification and mapping of anomalies, such as weeds, as well as the extraction of geometric measurements, such as plant volume/height estimates, will be integrated to enable precision farming practices. This would also improve the cost-benefit ratio of the sensor suite.

Chapter 5

Modelling the prediction-update loop

This section of the thesis is based on the published paper [126]. In this work, we propose a Neural Network to improve long-term state predictions without measurements based on Kalman filter observations. It is well known that the Kalman Filter is an iterative algorithm composed of two phases: predict and update. The update corrects predictions based on measurements. Predictions rely exclusively on the embedded physical model. This research aims to learn the underlying dynamics of the system under observation from the estimates of a standard Kalman Filter that supervises a Neural Network. Then, the Kalman Supervised Net (KSN) can be used to improve predictions learning from Kalman filter corrections. Numerical results show the advantages of the proposed solution when predicting the state of a spring-mass-damper system without using acceleration measurements.

5.1 Problem Statement

For the sake of simplicity, we refer to the stochastic Linear Time Invariant (LTI) dynamical system described in Eq. (5.1.0.1). However, the proposed approach can be extended to any complex system.

$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{F}\mathbf{x} + \mathbf{G}\mathbf{u} + \mathbf{\Gamma}\mathbf{w} \\ \mathbf{z} &= \mathbf{H}\mathbf{x} + \mathbf{D}\mathbf{u} + \mathbf{n}\end{aligned}\quad (5.1.0.1)$$

In Eq. (5.1.0.1), $\mathbf{x} \in \mathbb{R}^n$ is the state of the system, $\mathbf{u} \in \mathbb{R}^m$ the input, $\mathbf{z} \in \mathbb{R}^p$ the measurement vector, $\mathbf{w} \sim \mathcal{N}(0, \mathbf{Q})$ the process noise, and $\mathbf{n} \sim \mathcal{N}(0, \mathbf{R})$ the measurement noise. Moreover, $\mathbf{Q} \in \mathbb{R}^{n \times n}$ is the process noise covariance matrix, $\mathbf{R} \in \mathbb{R}^{p \times p}$ is the measurement noise covariance matrix and $\mathbf{\Gamma}$ is the matrix associated with the process noise. We consider the process noise and the measurement noise independent one each other and internally orthogonal, therefore \mathbf{Q} and \mathbf{R} are diagonal matrices.

Practical applications need a discrete-time version of the system (5.1.0.1), because

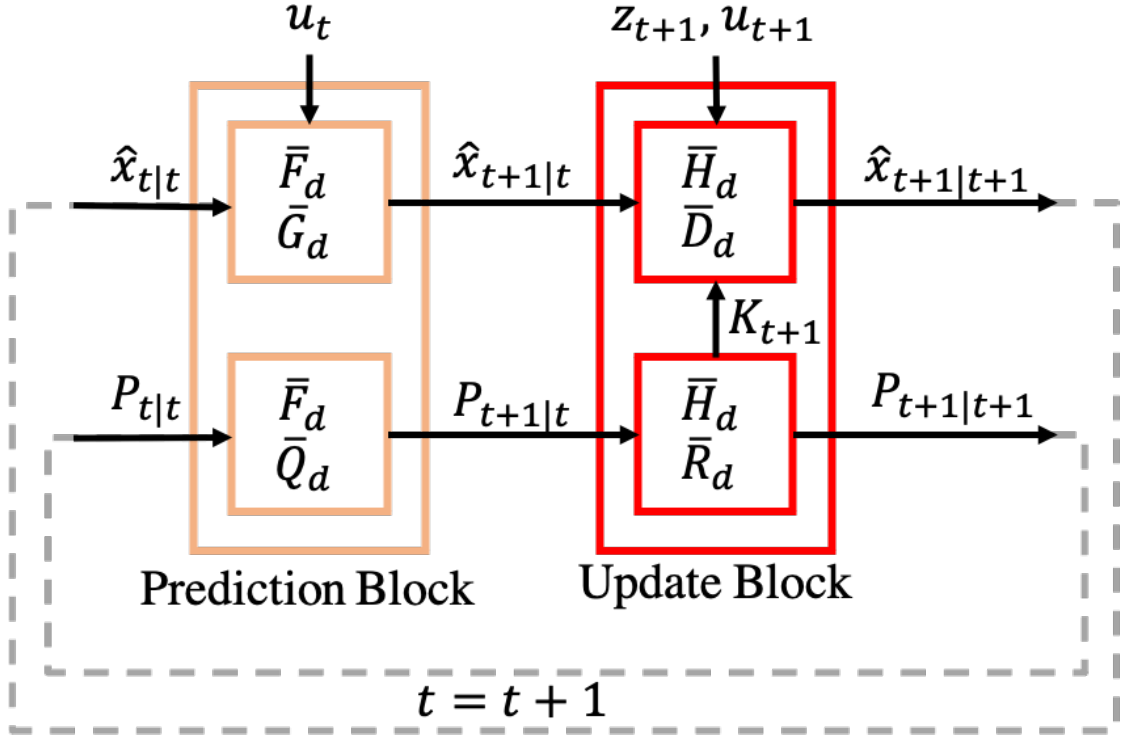


Figure 5.1.0.1: Block diagram representing the Kalman Filter whose plant model is characterized by matrices \bar{F}_d , \bar{G}_d , \bar{H}_d , \bar{D}_d , measurement covariance matrix \bar{R}_d and process noise covariance \bar{Q}_d . Here $\hat{x}_{t|t}$ is the posterior estimate at time t , u_t is the control input and $x_{t+1|t}$ is the state prediction at time $t + 1$ prior to measurement z_{t+1}

measurements are available only at the sampling points at each time step:

$$\mathbf{x}_{t+1} = \mathbf{F}_d \mathbf{x}_t + \mathbf{G}_d \mathbf{u}_t + \mathbf{\Gamma}_d \mathbf{w}_t \quad (5.1.0.2a)$$

$$\mathbf{z}_t = \mathbf{H}_d \mathbf{x}_t + \mathbf{D}_d \mathbf{u}_t + \mathbf{n}_t \quad (5.1.0.2b)$$

where $t \in \mathbb{N}$ the t -th time step. With Eq.s (5.1.0.2), we can define a suitable observer $\hat{\mathbf{x}}_t = \text{Obs}(\mathbf{z}_t, \hat{\mathbf{x}}_{t-1})$ able to estimate the state of system in Eq. (5.1.0.1) on the basis of the measurement \mathbf{z}_t and the previous observation $\hat{\mathbf{x}}_{t-1}$.

The Kalman filter is an iterative algorithm composed by two phases, namely, *prediction* and *update*, as explained in Figure 5.1.0.1. The *a priori* estimate $\hat{\mathbf{x}}_{t+1|t}$ is computed with a dynamical model of the plant that resembles Eq. (5.1.0.2a). In practical applications, the model embedded in the KF uses an approximation of the actual matrices that govern the dynamic process, as they are seldom known beforehand. The approximated matrices are indicated in Figure 5.1.0.1 with a bar, i.e., \bar{F}_d , \bar{G}_d , \bar{H}_d , \bar{D}_d . The KF is a robust algorithm for estimating system states whose evolution in time is corrupted

by noise, unmodelled phenomena and/or uncertainties in system parameters. The algorithm uses available measurements to correct *a priori* model-based predictions resulting in a better estimate of the state evolution over time. Therefore the purpose of this work is to construct a NN, which fits the hidden state evolution function of Eq. (5.1.0.2a) that the Kalman Filter emulates in a sequence of predictions and updates, from the knowledge of the matrices reported in Figure 5.1.0.1.

5.2 KSN usage

The sequence of state estimates obtained from a Kalman Filter working under nominal conditions is used to train the KSN. The network is trained using as input the previous posterior estimate $\hat{\mathbf{x}}_{t|t}$ together with the control input u_t at the same time, and it gives as output the posterior estimate $\hat{\mathbf{x}}_{t+1|t+1}$ at the successive time. The training aims at capturing the underlying dynamics of the system to improve prediction accuracy on state vector concerning the physics model embedded with the KF. The training stage of the neural predictor is shown in Figure 5.2.0.1 (a). It should be noted that the knowledge of the ground truth system dynamics is not used to train the KSN as this kind of information is supposed to be hidden and not available during in-field actual tests. The only required condition to train the KSN is that the Kalman Filter algorithm has converged and is properly working correcting physical predictions with the available measurements.

The network is trained with the collected data in an end-to-end manner, considering each time instant separately and shuffling the data in the mini-batches after every training epoch to ensure that only information regarding time instant t is needed to infer the state at time $t + 1$. Furthermore, the loss function defined as:

$$\mathcal{L} = \frac{1}{T} \sum_{t=1}^T \|\tilde{\mathbf{x}}_t - \hat{\mathbf{x}}_{t|t}\|^2 \quad (5.2.0.1)$$

being the MSE between network predictions $\tilde{\mathbf{x}}_t$ and KF posterior observations, is minimized on the training set ($T = \text{mini-batch size}$) and also computed on a validation set ($T = \text{validation size}$) extracted from the training data to impose a stopping condition for the training process thus avoiding over fitting. The trained network encapsulates system dynamics as a black-box built over the best available physical understanding of the actual system.

The Kalman Supervised Network can be designed with different architectures (RNN, CNN, etc..) depending on the dynamics complexity and the state dimension and can learn from different online observing algorithms (EKF, UKF, etc...) depending on both system characteristics and the detail level of the available physical description. The testing stage of the proposed algorithm is shown in Figure 5.2.0.1 (b). At time \bar{t} measurements are hidden and the KF embedded physical model predictions are propagated alongside KSN predictions.

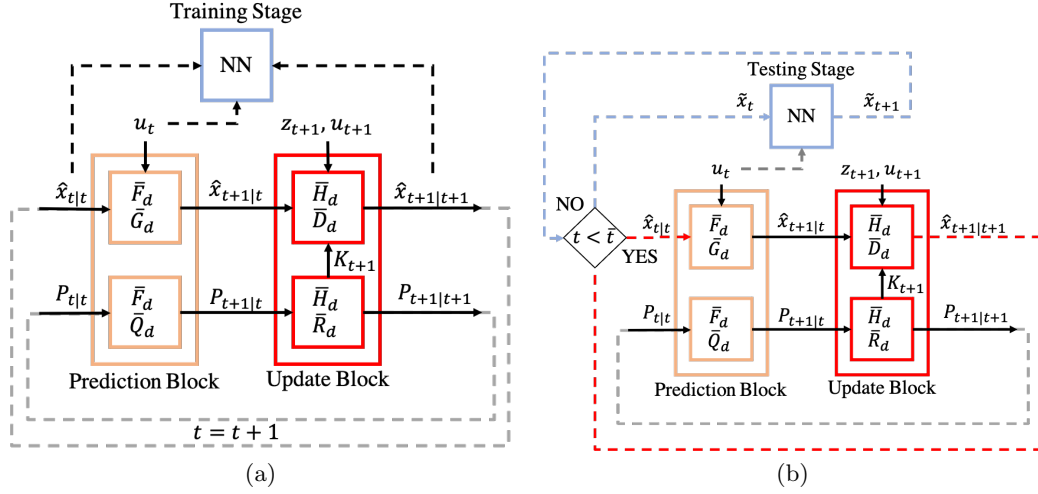


Figure 5.2.0.1: (a) Block diagram representing the training stage of the Kalman Supervised Network (b) block diagram representing the proposed estimation scheme. Time \bar{t} marks the beginning of the measurement outage when predictions of KSN are propagated alongside KF embedded physical model predictions. \tilde{x}_t refers to the network prediction at time $t > \bar{t}$.

5.3 Numerical Results

The proposed KSN is applied to the well-known mass-spring-damper system. Firstly, the ground-truth system is introduced. Then, a discrete-time Kalman Filter observer is defined grounded on an approximate model. Moreover, a Neural Network for implementing the Kalman Supervised Network is designed and trained on Kalman Filter predictions for a sinusoidal input signal. Finally, the Neural Network has been tested for a different input.

5.3.1 The mass-spring-damper system

Figure 5.3.1.1 shows the classical mass-spring-damper system, where m , k , and c represent, respectively, the mass, the stiffness and the damping coefficient.

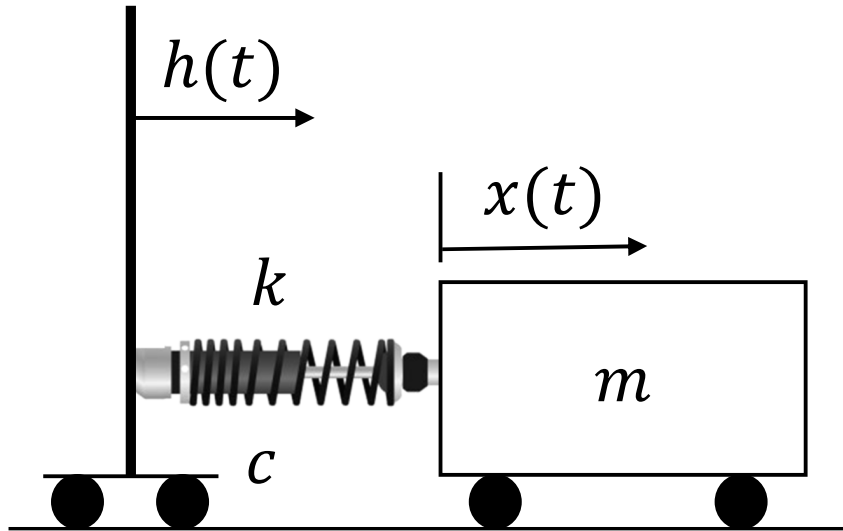


Figure 5.3.1.1: Mass-spring-damper system, where m is the mass of the system, k is the elastic coefficient of the spring, c is the damping coefficient, $h(t)$ is the input and $x(t)$ is the mass displacement

The dynamics of the system can be modeled by the following second-order Ordinary Differential Equation (ODE):

$$m\ddot{x} + c(\dot{x} - \dot{h}) + k(x - h) = 0 \quad (5.3.1.1)$$

Then, a state-space representation can be obtained by selecting the following state variables

$$\mathbf{x} = \begin{bmatrix} x - h \\ \dot{x} \end{bmatrix} \quad (5.3.1.2)$$

with the input to the system given by $\mathbf{u} = \dot{h}$. With reference to Eqs. (5.1.0.1), let us assume that the acceleration measurements of the mass m are available, *i.e.* $\mathbf{z} = \ddot{x}$, then, the following matrices are defined:

$$\mathbf{F} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix} -1 \\ \frac{c}{m} \end{bmatrix} \quad (5.3.1.3)$$

$$\mathbf{H} = \begin{bmatrix} -\frac{k}{m} & -\frac{c}{m} \end{bmatrix}, \quad \mathbf{D} = \frac{c}{m}.$$

Clearly, the parameters of the model are unknown during field tests. Furthermore, measurements are corrupted by both process and measurement noise. For this reason, the process and measurement input matrices, respectively \mathbf{G} and \mathbf{D} , are modified as follows

$$\mathbf{G}_w = \begin{bmatrix} -1 & 1 & 0 & 0 \\ \frac{c}{m} & 0 & 1 & 0 \end{bmatrix} \quad (5.3.1.4)$$

$$\mathbf{D}_w = \begin{bmatrix} \frac{c}{m} & 0 & 0 & 1 \end{bmatrix}$$

and, consequently, the input is modified to account for three different additive Gaussian noises:

$$\mathbf{u}_w = [\dot{h} \quad w_1 \quad w_2 \quad w_a]^T \quad (5.3.1.5)$$

where $w_i \sim \mathcal{N}(0, \sigma_i^2)$ for $i = 1, 2, a$, is the noise associated to the first state, second state, and acceleration measurements, respectively.

The time evolution of the state used as ground truth reference for the tests is obtained using the noiseless state-space model with \mathbf{G} , \mathbf{D} and \mathbf{u} , while measurements are extracted from the noisy state-space model with \mathbf{G}_w , \mathbf{D}_w and \mathbf{u}_w . The parameter values selected for numerical simulations are listed in Table 5.3.1.1, while Table 5.3.1.2 contains the selected noises associated with process and measurements.

Table 5.3.1.1: True system parameters

Parameter	Value	Unit of Measurement
m	100	kg
k	10000	N/m
c	200	Ns/m

5.3.2 KF Implementation

Let us assume that the parameters in Table 5.3.1.1 are known with a confidence interval of 5%. Moreover, we assume to have no information about noise covariance matrices. Therefore, the approximated model embedded in the KF is built upon the matrices $\bar{\mathbf{F}}_d$, $\bar{\mathbf{G}}_d$, $\bar{\mathbf{H}}_d$ and $\bar{\mathbf{D}}_d$ that are constructed following Eq. (5.3.1.3) with the parameters affected with a random percentage error up to 5%. Meanwhile, covariance matrices $\bar{\mathbf{Q}}_d$ and $\bar{\mathbf{R}}_d$ contain twice the value of the variances in Table 5.3.1.2. Finally, the discrete time model is achieved by means of the forward Euler integration method [108].

Table 5.3.1.2: True noise variances

Parameter	Value	Unit of measure
σ_1^2	0.1	(m) ²
σ_2^2	1	(m/s) ²
σ_a^2	0.35	(m/s ²) ²

5.3.3 KSN Training

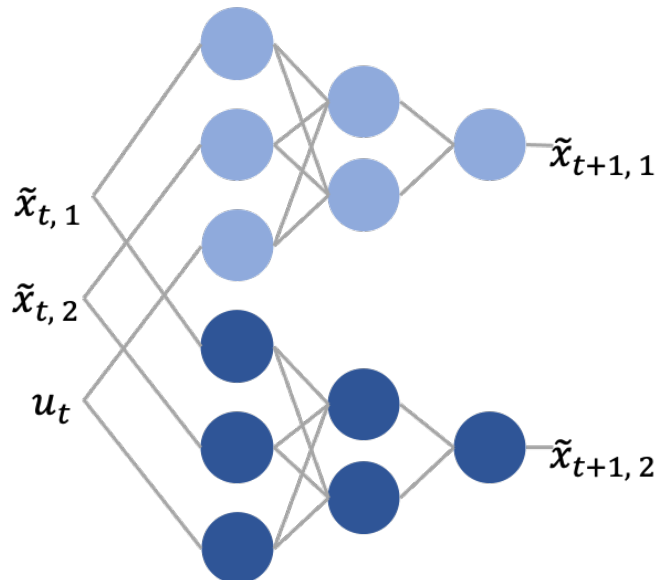


Figure 5.3.3.1: Kalman Supervised Network layout used for the considered mechanical system

Two distinct networks are considered with a feed-forward structure to output separately predictions on the two state variables. The training process, already introduced in Sec. 5.2, is therefore carried out separately for each one of the two components of the state x . We recall from Sec. 5.3.2 that training examples are obtained from the *a posteriori* estimates of the Kalman Filter observer exploiting measurements. The two networks present the same architecture with one sequence input layer followed by a fully-connected hidden layer and a regression output layer. A feed-forward architecture has been selected to characterize time independence of this system [1, 11]. A graphical representation of the structure of the network is depicted in Fig. 5.3.3.1. The training

set is extracted from the observations obtained by the linear Kalman filter when the system is fed with a sinusoidal input of the form $\mathbf{u} = \dot{h} = 2\pi \sin(2\pi t + \pi/2)$ for a time interval of 5 seconds, corresponding to a constrained training displacement $h = \sin(2\pi t)$. The training process is carried on with a mini-batch containing 2% of the available training samples and stops considering the loss computed on a validation set with 15% of the training data. For each training iteration the loss in Eq. (5.2.0.1) is minimized inside the mini-batch, updating net parameters and after 50 iterations the validation loss is computed in the validation set to check for the stopping condition. The training stops when the validation loss remains larger or equal to the smallest one computed for 50 consecutive validation checks. For a sinusoidal input on a time interval of 5 s, the training process takes around 25 seconds for each one of the two networks composing the described KSN. The training process has been carried on an intel i-9 CPU under MATLAB environment.

5.3.4 KSN Testing

The Neural Predictor is tested on the input signal expressed by Eq. (5.3.4.1):

$$\dot{h}(t) = \sum_{i=1}^N \frac{d}{dt} \left[\sqrt{2\Phi_i(\omega_i)\Delta\omega} \sin(\omega_i t - \varphi_i) \right] \quad (5.3.4.1)$$

where ω_i is the i -th angular frequency, Φ_i is the Power Spectral Density (PSD) computed at ω_i , $\Delta\omega$ is the angular frequency step size and φ_i is the i -th phase angle belonging to the random vector $\boldsymbol{\varphi} \sim \mathcal{N}(0, (2\pi)^2)$. For the testing stage we used a signal composed of $N = 10000$ sinusoidal waves from $\omega_1 = 2\pi$ to $\omega_N = 200\pi$. It is important to note that testing is performed by predicting the system response when excited with a different input than that used for the training stage. This is a more challenging task since we evaluate the ability of the system to extrapolate on a previously unseen population that is independent from the training dataset.

Figure 5.3.4.1 shows the testing results for the aforementioned input signal. It is assumed that measurements are available for the first 3 s of the simulation ($t < \bar{t}$, $\bar{t} = 3$ s) when the standard Kalman Filter (solid red line) correctly provides state estimates matching the ground truth model (dashed black line). The good agreement between KF observations and ground truth data underlines the KF correct working conditions. At time $\bar{t} = 3$ s, mass acceleration measurements are made unavailable to test the proposed algorithm, and KF predictions are propagated in time to keep assessing system state relying only on the embedded model. During the measurement outage stage, the KF predictions are compared with those provided by the KSN (solid blue line). As seen from Figure 5.3.4.1. In order to quantitatively evaluate the system performance, suitable metrics are considered. Given the RMSEs that measure the discrepancy of a given

observer with respect to ground truth data

$$\begin{aligned}
\text{RMSE}_{N,i} &= \sqrt{\frac{1}{T} \sum_{t=\bar{t}}^T (\tilde{\mathbf{x}}_{t,i} - \mathbf{x}_{t,i})^2} \\
\text{RMSE}_{K,i} &= \sqrt{\frac{1}{T} \sum_{t=\bar{t}}^T (\hat{\mathbf{x}}_{t|t-1,i} - \mathbf{x}_{t,i})^2} \\
\text{RMSE}_{B,i} &= \sqrt{\frac{1}{T} \sum_{t=\bar{t}}^T (\hat{\mathbf{x}}_{t|t,i} - \mathbf{x}_{t,i})^2}
\end{aligned} \tag{5.3.4.2}$$

where RMSE_N refers to the proposed KSN, RMSE_K is obtained from $\hat{\mathbf{x}}_{t|t-1}$ that represents the *a priori* physical-based prediction at time index t . Finally, RMSE_B refers to the benchmark error as obtained from a virtual KF that keeps receiving in input the measurements. Index $i = 1, 2$ in Eq. (5.3.4.2) refers to the two state variables of the system. Then, three evaluation metrics can be considered:

- ψ_N : the relative percentage variation in the RMSE between the state predictions obtained from the KSN and the KF.
- Δ_N : the absolute variation in the RMSE between RMSE_N and RMSE_B
- Δ_K : the absolute variation in the RMSE between RMSE_K and RMSE_B .

The mathematical expression of the three evaluation metrics ψ_N , Δ_N and Δ_K is detailed in Eq. (5.3.4.3). A negative value of $\psi_{N,i}$ indicates that consecutive predictions of the KSN are more accurate than consecutive KF predictions. Positive values are instead expected for $\Delta_{K,i}$, because the output of a correctly working KF should be the closest estimate available of the ground truth state evolution. Similarly, positive values are expected for $\Delta_{N,i}$ because the KSN is trained over KF posterior observations, so it will not be able to return RMSE values smaller than those obtained with the observer from which it learns.

$$\begin{aligned}
\psi_{N,i} &= \frac{\text{RMSE}_{N,i} - \text{RMSE}_{K,i}}{\text{RMSE}_{K,i}} \times 100 \\
\Delta_{N,i} &= \text{RMSE}_{N,i} - \text{RMSE}_{B,i} \\
\Delta_{K,i} &= \text{RMSE}_{K,i} - \text{RMSE}_{B,i}
\end{aligned} \tag{5.3.4.3}$$

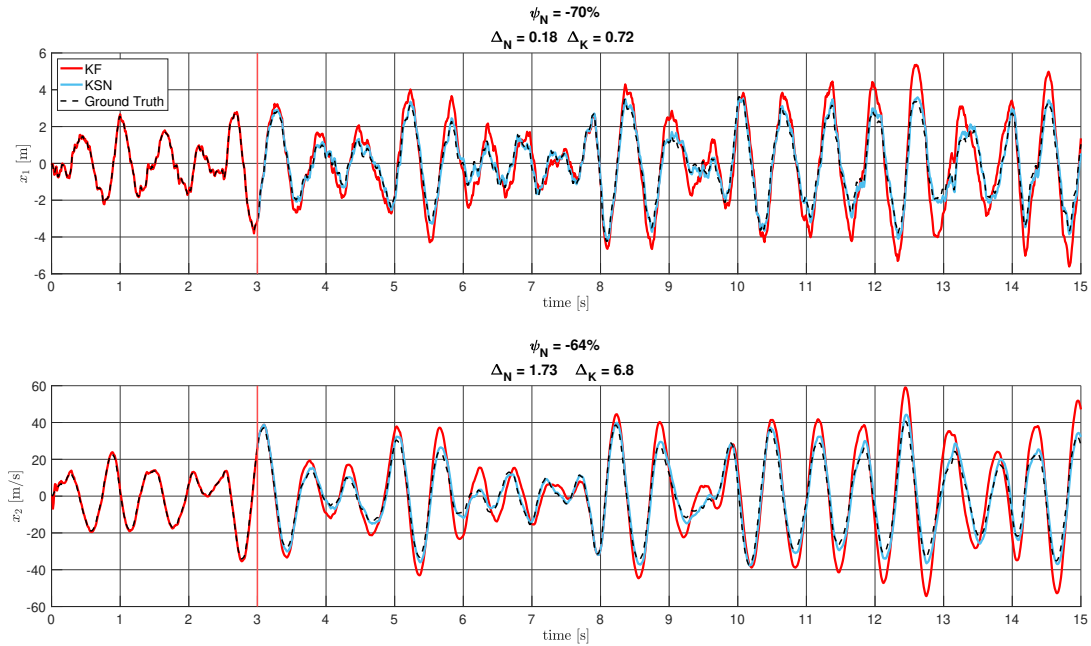


Figure 5.3.4.1: Testing results obtained by comparing state estimation as obtained from the KSN and the standard KF during measurement outage starting at time $t = 3$ s

Comparison between the proposed prediction algorithm KSN and the standard KF without measurement updates is shown in Figure 5.3.4.1, in terms of both evaluation metrics and time history of the state evolution. The KSN clearly outperforms the KF in predicting the system evolution over time. For the first state variable, the $RMSE_N$ is reduced of 70%, whereas the improvement is of 64% for the second one. For both state variables, it holds true the relationship $0 < \Delta_{N,i} < \Delta_{K,i}$ indicating that the KF observer provides the best state estimates exploiting measurements, and that KSN predictions are more accurate than KF ones when propagated over long-time periods. Although the results have been obtained from the textbook use case of a mass-spring-damper system, they are promising as well as more complex systems.

The RMSE reduction has been confirmed for a 12 s long sensor signal shortage, and the KSN only required a 5 s long observation for a sinusoidal input and 50 s of training process. The only condition needed for the KSN training is the reliability of KF observations that can be assessed in several ways. For the simulated case at hand, reliability was ensured by the knowledge of the ground truth system dynamics. On real word applications it can be inferred analysing the distribution of innovation vectors [114] or other metrics such those discussed in [50].

Considerations The presented algorithm has dealt with the problem of improving long-term predictions for a dynamic system using Kalman Filter observations. Neural networks are widely used to fit generic functions by learning relationships between inputs and outputs directly from collected data. The KSN algorithm is based on the assumption that the real dynamics of a generic system is inherently unknown and can only be estimated through an observer that combines frequent measurements with the best physical understanding of the phenomenon. Therefore the KSN is trained on reliable observations to build an accurate predictor that learns from the information provided by the measurements and captured system dynamics providing a transfer function more accurate than the one implemented in the KF prediction block. The KSN algorithm has been explained and tested on a mass-spring-damper system where it showed to be more accurate than a standard KF for long-term predictions without measurements. In future works the proposed method will be applied to more complex systems and validated with in-field tests. In addition, recurrent network architectures and possible advantages of transfer learning for these data driven models will be investigated.

Chapter 6

Recommendations for future work

This thesis presented evidence of the effectiveness of machine learning and deep learning in mechanical engineering, particularly in terrain classification. The comparison of Convolutional Neural Networks and Recurrent Neural Networks to standard machine learning techniques, such as Support Vector Machines, highlighted the advantage of Deep Learning in modeling complex phenomena where traditional methods were insufficient. The proposed Multichannel Spectrograms led terrain classification accuracy of over 90% using a CNN based on proprioceptive signals demonstrating the capability of Deep Learning in overcoming both generalization and extrapolation problems. The use of exteroceptive measures to predict wheel-terrain interactions and the examination of a combination of proprioceptive and exteroceptive signals for crop monitoring reinforced the benefits of Deep Learning in enhancing the performance of rovers and control systems. The novel Kalman Supervised Network algorithm, which continuously learns from sensor measurements to produce a more accurate model of a mechanical system than the one embedded in the Kalman Filter, further highlighted the potential of Deep Learning in mechanical engineering.

Future work will focus on the usage of KSN for enhancing control strategies. Supposing, for example to use a simple Proportional control for a given system described by the state transition matrix A and the control matrix B , to reach the desired state x_g the applied control u_t at time instant t would be

$$u_t = K(x_g - x_{t|t}) \quad (6.0.0.1)$$

where $x_{t|t}$ is the Kalman Filter state estimate at time t . The problem is that every control strategy does not learn from measures corrections but simply adapts the control based on how far away the system is from the goal x_g . The computed control u_t is supposed to make the system pass to state $x_{t+1|t}$ as it is only tuned with knowledge of matrixes A and B . What will actually happen is that provided u_t the system will transition to the posterior estimated state $x_{t+1|t+1}$ and the subsequent control will be based on that corrected estimate.

While this reasoning will eventually lead the system to the desired goal x_g the sequence of state transitions will not be the one initially prescribed when the proportional

control matrix K was designed. KSN instead has the main purpose of learning the system dynamics from the Kalman Filter so it provides an accurate mathematical representation of the system learned from measures corrections. It is necessary here to underline the concept that the Kalman Filter does not learn from the corrections that it makes to physics predictions exploiting available measures, but instead it uses the measures to correct the estimate at each instant t . KSN instead is specifically designed to learn from the dynamic that KF represents in a loop of predictions and corrections, therefore outputs a generic non-linear function $f(x, u)$ that encapsulates the actual dynamic of the system.

The intended future use of KSN is to compute the control u_t that will take the system to state $x_{t+1|t}$ that the designed control was supposed to achieve. For such scope, one will need to solve the non-linear equation:

$$Ax_{t|t} + BK(x_g - x_{t|t}) = f(x_{t|t}, u_t) \quad (6.0.0.2)$$

for the control u_t . Considering that if the activation functions used in KSN are continuous and differentiable, then also function f is continuous and differentiable, this enables iterative methods for solving non-linear equations, such as Newton-Raphson, to solve around $BK(x_g - x_{t|t})$ for the control u_t . If KSN is correctly trained over a functioning KF, this solution will take the system to the desired state following the desired path avoiding systematic errors due to the fact that the control strategy does not learn from measures corrections but only adapts to them.

Appendix A

Author's Publications

The following articles have been produced with a significant contribution of the Author:

Ashwin Rajkumar, Fabio Vulpi, Satish Reddy Bethi, Hassam Khan Wazir, Preeti Raghavan, and Vikram Kapila. Wearable inertial sensors for range of motion assessment. *IEEE sensors journal*, 20(7):3777–3787, 2019

Ashwin Rajkumar, Fabio Vulpi, Satish Reddy Bethi, Preeti Raghavan, and Vikram Kapila. Usability study of wearable inertial sensors for exergames (wise) for movement assessment and exercise. *Mhealth*, 7, 2021

Satish Reddy Bethi, Ashwin Rajkumar, Fabio Vulpi, Preeti Raghavan, and Vikram Kapila. Wearable inertial sensors for exergames and rehabilitation. In *2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pages 4579–4582. IEEE, 2020

F Vulpi, A Milella, F Cordes, R Dominguez, and G Reina. Deep terrain estimation for planetary rovers. In *15th International Symposium on Artificial Intelligence, Robotics and Automation in Space, ISAIRAS-2020*, 2020

Angelo Ugenti, Fabio Vulpi, Annalisa Milella, and Giulio Reina. Learning and prediction of vehicle-terrain interaction from 3d vision. In *Multimodal Sensing and Artificial Intelligence: Technologies and Applications II*, volume 11785, pages 167–173. SPIE, 2021

Antonio Petitti, Fabio Vulpi, Roberto Marani, and Annalisa Milella. A self-calibration approach for multi-view RGB-D sensing. In Ettore Stella, editor, *Multimodal Sensing and Artificial Intelligence: Technologies and Applications II*, volume 11785, pages 50 – 55. International Society for Optics and Photonics, SPIE, 2021

Fabio Vulpi, Annalisa Milella, Roberto Marani, and Giulio Reina. Recurrent and convolutional neural networks for deep terrain classification by autonomous robots. *Journal of Terramechanics*, 96:119–131, 2021

Angelo Ugenti, Fabio Vulpi, Raúl Domínguez, Florian Cordes, Annalisa Milella, and Giulio Reina. On the role of feature and signal selection for terrain learning in planetary exploration robots. *Journal of Field Robotics*, 39(4):355–370, 2022

Fabio Vulpi, Roberto Marani, Antonio Petitti, Giulio Reina, and Annalisa Milella. An rgb-d multi-view perspective for autonomous agricultural robots. *Computers and Electronics in Agriculture*, 202:107419, 2022

Fabio Vulpi, Antonio Leanza, Antonio Petitti, Annalisa Milella, and Giulio Reina. Kalman supervised network for improved model predictions. In *2022 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*, pages 1–7. IEEE, 2022

Arianna Rana, Fabio Vulpi, Rocco Galati, Annalisa Milella, and Antonio Petitti. A pose estimation algorithm for agricultural mobile robots using an rgb-d camera. In *2022 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*, pages 1–5, 2022

References

- [1] Manuel Acosta and Stratis Kanarachos. Tire lateral force estimation and grip potential identification using neural networks, extended kalman filter, and recursive least squares. *Neural Computing and Applications*, 30(11):3445–3465, 2018.
- [2] J Aguilar, Alberto Garces-Jimenez, MD R-Moreno, and Rodrigo García. A systematic literature review on the use of artificial intelligence in energy self-management in smart buildings. *Renewable and Sustainable Energy Reviews*, 151:111530, 2021.
- [3] Joelle Al Hage, Stefano Mafrika, Maan El Badaoui El Najjar, and Franck Ruffier. Informational framework for minimalistic visual odometry on outdoor robot. *IEEE Transactions on Instrumentation and Measurement*, 68(8):2988–2995, 2018.
- [4] A. Angelova, L. Matthies, D. Helmick, and P. Perona. Learning and prediction of slip. *Journal of Field Robotics*, 24:205–231, 2006.
- [5] Anelia Angelova, Larry Matthies, Daniel Helmick, and Pietro Perona. Learning and prediction of slip from visual information. *Journal of Field Robotics*, 24(3):205–231, 2007.
- [6] Chengchao Bai, Jifeng Guo, Linli Guo, and Junlin Song. Deep multi-layer perception based terrain classification for planetary exploration rovers. *Sensors*, 19, 2019.
- [7] Dan Barnes, Will Maddern, and Ingmar Posner. Find your own way: Weakly-supervised segmentation of path proposals for urban autonomy. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 203–210. IEEE, 2017.
- [8] Jakub Bednarek, Michal Bednarek, Lorenz Wellhausen, Marco Hutter, and Krzysztof Walas. What am i touching? learning to classify terrain via haptic sensing. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7187–7193. IEEE, 2019.
- [9] Jörg Behler. Neural network potential-energy surfaces in chemistry: a tool for large-scale simulations. *Physical Chemistry Chemical Physics*, 13(40):17930–17955, 2011.

-
- [10] M. Bekker. *Introduction to Terrain-Vehicle Systems*. University of Michigan Press, Ann Arbor, 1969.
- [11] Igor Belič. Neural networks and static modelling. *Recurrent neural networks and soft computing*, page 1, 2012.
- [12] Mauro Bellone, Giulio Reina, Luca Caltagirone, and Mattias Wahde. Learning traversability from point clouds in challenging scenarios. *IEEE Transactions on Intelligent Transportation Systems*, 19:296–305, 2018.
- [13] P Bellutta, R Manduchi, L Matthies, K Owens, and A Rankin. Terrain perception for demo iii. *IEEE Intelligent Vehicles Symposium*, pages 326–331, 2000.
- [14] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.
- [15] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):349–359, 1999.
- [16] Satish Reddy Bethi, Ashwin RajKumar, Fabio Vulpi, Preeti Raghavan, and Vikram Kapila. Wearable inertial sensors for exergames and rehabilitation. In *2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pages 4579–4582. IEEE, 2020.
- [17] L Breiman. Random forests mach learn 45 (1): 5–32, 2001.
- [18] Christopher A Brooks and Karl Iagnemma. Vibration-based terrain classification for planetary exploration rovers. *IEEE Transactions on Robotics*, 21(6):1185–1191, 2005.
- [19] Christopher A Brooks and Karl Iagnemma. Self-supervised terrain classification for planetary surface exploration rovers. *Journal of Field Robotics*, 29(3):445–468, 2012.
- [20] Mustafa Cakir, Mehmet Ali Guvenc, and Selcuk Mistikoglu. The experimental application of popular machine learning algorithms on predictive maintenance and the design of iiot based condition monitoring system. *Computers & Industrial Engineering*, 151:106948, 2021.
- [21] Yann Chéné, David Rousseau, Philippe Lucidarme, Jessica Bertheloot, Valérie Caffier, Philippe Morel, Étienne Belin, and François Chapeau-Blondeau. On the use of depth camera for 3d phenotyping of entire plants. *Computers and Electronics in Agriculture*, 82:122–127, 2012.
- [22] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, and S. Thrun. *Principles of Robot Motion*. The MIT Press, Cambridge, MA, USA, 2004.

- [23] Lorenzo Comba, Alessandro Biglia, Davide Ricauda Aimonino, and Paolo Gay. Unsupervised detection of vineyards by 3d point-cloud uav photogrammetry for precision agriculture. *Computers and Electronics in Agriculture*, 155:84–95, 2018.
- [24] Isabella CFS Condotta, Tami M Brown-Brandl, Santosh K Pitla, John P Stinn, and Késia O Silva-Miranda. Evaluation of low-cost depth cameras for agricultural applications. *Computers and Electronics in Agriculture*, 173:105394, 2020.
- [25] Florian Cordes, Ajish Babu, and Frank Kirchner. Static force distribution and orientation control for a rover with an actively articulated suspension system. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5219–5224. IEEE, 2017.
- [26] Florian Cordes, Frank Kirchner, and Ajish Babu. Design and field testing of a rover with an actively articulated suspension system in a mars analog terrain. *Journal of Field Robotics*, 35(7):1149–1181, 2018.
- [27] Massimiliano Corsini, Paolo Cignoni, and Roberto Scopigno. Efficient and flexible sampling with blue noise properties of triangular meshes. *IEEE Transactions on Visualization and Computer Graphics*, 18(6):914–924, 2012.
- [28] R Cowen. Opportunity rolls out of purgatory. *Science News*, 167(26):413, 2005.
- [29] Patrick Dallaire, Krzysztof Walas, Philippe Giguere, and Brahim Chaib-draa. Learning terrain types with the pitman-yor process mixtures of gaussians for a legged robot. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3457–3463. IEEE, 2015.
- [30] Rosa Pia Devanna, Annalisa Milella, Roberto Marani, Simone Pietro Garofalo, Gaetano Alessandro Vivaldi, Simone Pascuzzi, Rocco Galati, and Giulio Reina. In-field automatic identification of pomegranates using a farmer robot. *Sensors*, 22(15):5821, 2022.
- [31] Thomas G Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of artificial intelligence research*, 2:263–286, 1994.
- [32] Mauro Dimastrogiovanni, Florian Cordes, and Giulio Reina. Terrain estimation for planetary exploration robots. *Applied Sciences*, 10(17):6044, 2020.
- [33] Kai-Bo Duan and S Sathiya Keerthi. Which is the best multiclass svm method? an empirical study. In *International workshop on multiple classifier systems*, pages 278–285. Springer, 2005.
- [34] Edmond M DuPont, Carl A Moore, Emmanuel G Collins Jr, and Eric J Coyle. Frequency response method for online terrain identification in unmanned ground vehicles. *Autonomous Robots*, 24(4):337–347, 2008.

- [35] Tatjana Eitrich and Bruno Lang. Efficient optimization of support vector machine learning parameters for unbalanced datasets. *Journal of computational and applied mathematics*, 196(2):425–436, 2006.
- [36] ESA. Esa exploration. <http://exploration.esa.int/mars/>. Accessed: 2023-2-28.
- [37] EU. Ade project. <https://www.h2020-ade.eu/>. Accessed: 2023-2-28.
- [38] EU. Peraspera. <http://www.h2020-peraspera.eu/>. Accessed: 2023-2-28.
- [39] Alberto Gallina, Rainer Krenn, Marco Scharringhausen, Tadeusz Uhl, and Bernd Schäfer. Parameter identification of a planetary rover wheel–soil contact model via a bayesian approach. *Journal of Field Robotics*, 31(1):161–175, 2014.
- [40] Gazebo. Gazebo simulator. <https://gazebo.org/home>. Accessed: 2023-2-28.
- [41] Michael A Gelbart, Jasper Snoek, and Ryan P Adams. Bayesian optimization with unknown constraints. *arXiv preprint arXiv:1403.5607*, 2014.
- [42] P Giguere and G Dudek. Clustering sensor data for autonomous terrain identification using time-dependency. *Autonomous Robots*, 26:171–186, 2009.
- [43] Steven B Goldberg, Mark W Maimone, and Larry Matthies. Stereo vision and rover navigation software for planetary exploration. In *Proceedings, IEEE aerospace conference*, volume 5, pages 5–5. IEEE, 2002.
- [44] A Gongal, Suraj Amatya, Manoj Karkee, Q Zhang, and Karen Lewis. Sensors and systems for fruit detection and localization: A review. *Computers and Electronics in Agriculture*, 116:8–19, 2015.
- [45] Ramon Gonzalez, Samuel Chandler, and Dimi Apostolopoulos. Characterization of machine learning algorithms for slippage estimation in planetary exploration rovers. *Journal of Terramechanics*, 82:23–34, 2019.
- [46] Ramon Gonzalez and Karl Iagnemma. Deepterrmechanics: Terrain classification and slip estimation for ground robots via deep learning. *arXiv preprint arXiv:1806.07379*, 2018.
- [47] Ramon Gonzalez, A Rituerto, and J Guerrero. Improving robot mobility by combining downward-looking and frontal cameras. *Robotics*, 5:25–44, 2016.
- [48] Ramon Gonzalez, Chandler Samuel, and Dimi Apostolopoulos. Characterization of machine learning algorithms for slippage estimation in planetary exploration rovers. *Journal of Terramechanics*, 82:23–34, 2019.
- [49] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. Ieee, 2013.

-
- [50] Mohinder S Grewal and Angus P Andrews. *Kalman filtering: Theory and Practice with MATLAB*. John Wiley & Sons, 2014.
- [51] Lisa Grossman. Atom & cosmos: Spirit stuck, but in good spot. *Science News*, 177(5):7–7, 2010.
- [52] Junlong Guo, Tianyou Guo, Ming Zhong, Haibo Gao, Bo Huang, Liang Ding, Weihua Li, and Zongquan Deng. In-situ evaluation of terrain mechanical parameters and wheel-terrain interactions using wheel-terrain contact mechanics for wheeled planetary rovers. *Mechanism and Machine Theory*, 145:103696, 2020.
- [53] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.
- [54] Novian Habibie, Aditya Murda Nugraha, Ahmad Zaki Anshori, M Anwar Ma’sum, and Wisnu Jatmiko. Fruit mapping mobile robot on simulated agricultural area in gazebo simulator using simultaneous localization and mapping (slam). In *2017 International Symposium on Micro-NanoMechatronics and Human Science (MHS)*, pages 1–7. IEEE, 2017.
- [55] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.
- [56] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997.
- [57] Matej Hoffmann, Karla Štěpánová, and Michal Reinstein. The effect of motor action and different sensory modalities on terrain classification in a quadruped robot running with multiple gaits. *Robotics and Autonomous Systems*, 62(12):1790–1798, 2014.
- [58] Jeffrey Humpherys, Preston Redd, and Jeremy M. West. A fresh look at the kalman filter. *SIAM Rev.*, 54(4):801–823, 2012.
- [59] Karl Iagnemma, Shinwoo Kang, Hassan Shibly, and Steven Dubowsky. Online terrain parameter estimation for wheeled mobile robots with application to planetary rovers. *IEEE transactions on robotics*, 20(5):921–927, 2004.
- [60] Marco Imperoli, Ciro Potena, Daniele Nardi, Giorgio Grisetti, and Alberto Pretto. An effective multi-cue positioning system for agricultural robotics. *IEEE Robotics and Automation Letters*, 3(4):3685–3692, 2018.
- [61] Reina Ishikawa, Ryo Hachiuma, and Hideo Saito. Self-supervised audio-visual feature learning for single-modal incremental terrain type clustering. *IEEE Access*, 9:64346–64357, 2021.

- [62] Peng Jiang, Yuehan Chen, Bin Liu, Dongjian He, and Chunquan Liang. Real-time detection of apple leaf diseases using deep learning approach based on improved convolutional neural networks. *IEEE Access*, 7:59069–59080, 2019.
- [63] Xianjian Jin, Guodong Yin, and Nan Chen. Advanced estimation techniques for vehicle system dynamic state: A survey. *Sensors*, 19(19), 2019.
- [64] Rudolph Emile Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.
- [65] Fazle Karim, Somshubra Majumdar, Houshang Darabi, and Shun Chen. Lstm fully convolutional networks for time series classification. *IEEE access*, 6:1662–1669, 2017.
- [66] Hee-Un Kim and Tae-Suk Bae. Deep learning-based gnss network-based real-time kinematic improvement for autonomous ground vehicle navigation. *Journal of Sensors*, 2019:3737265, 2019.
- [67] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [68] Dmitrii Kochkov, Jamie A Smith, Ayya Alieva, Qing Wang, Michael P Brenner, and Stephan Hoyer. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21):e2101784118, 2021.
- [69] Fleur Legrain, Jesús Carrete, Ambroise van Roekeghem, Stefano Curtarolo, and Natalio Mingo. How chemical composition alone can predict vibrational free energies and entropies of solids. *Chemistry of Materials*, 29(15):6220–6227, 2017.
- [70] Hongfei Li, Dhaivat Parikh, Qing He, Buyue Qian, Zhiguo Li, Dongping Fang, and Arun Hampapur. Improving rail network velocity: A machine learning approach to predictive maintenance. *Transportation Research Part C: Emerging Technologies*, 45:17–26, 2014.
- [71] Zhenbo Li, Ruohao Guo, Meng Li, Yaru Chen, and Guangyao Li. A review of computer vision technologies for plant phenotyping. *Computers and Electronics in Agriculture*, 176:105672, 2020.
- [72] Yuran Liang, Steffen Müller, Daniel Rolle, Dieter Ganesch, and Immanuel Schaffer. Vehicle side-slip angle estimation with deep neural network and sensor data fusion. In *10th International Munich Chassis Symposium 2019*, pages 159–178. Springer, 2020.
- [73] Jacqueline Libby and Anthony J Stentz. Using sound to classify vehicle-terrain interactions in outdoor environments. In *2012 IEEE International Conference on Robotics and Automation*, pages 3559–3566. IEEE, 2012.

- [74] Yi Lin, Yoonkyung Lee, and Grace Wahba. Support vector machines for classification in nonstandard situations. *Machine learning*, 46(1):191–202, 2002.
- [75] Dengsheng Lu and Qihao Weng. A survey of image classification methods and techniques for improving classification performance. *International journal of Remote sensing*, 28(5):823–870, 2007.
- [76] Hai-Bang Ly, Thuy-Anh Nguyen, and Binh Thai Pham. Estimation of soil cohesion using machine learning method: A random forest approach. *Advances in civil engineering*, 2021:1–14, 2021.
- [77] Juncheng Ma, Keming Du, Lingxian Zhang, Feixiang Zheng, Jinxiang Chu, and Zhong yu Sun. A segmentation method for greenhouse vegetable foliar disease spots images using color information and region growing. *Comput. Electron. Agric.*, 142:110–117, 2017.
- [78] Jennifer Mack, Christian Lenz, Johannes Teutrine, and Volker Steinhage. High-precision 3d detection and reconstruction of grapes from laser range data for efficient phenotyping based on supervised learning. *Computers and Electronics in Agriculture*, 135:300–311, 2017.
- [79] R Manduchi, A Castano, A Talukder, and L Matthies. Obstacle detection and terrain classification for autonomous off-road navigation. *Autonomous Robots*, 18:81–102, 2005.
- [80] J Martinez-Gomez, A Fernandez-Cabellero, I Garcia-Varea, L Rodriguez, and C Romero-Gonzalez. A taxonomy of vision systems for ground mobile robots. *International Journal of Advanced Robotic Systems*, 11:1–26, 2014.
- [81] Alessandro Matese, Piero Toscano, Salvatore Filippo Di Gennaro, Lorenzo Genesio, Francesco Primo Vaccari, Jacopo Primicerio, Claudio Belli, Alessandro Zaldei, Roberto Bianconi, and Beniamino Gioli. Intercomparison of uav, aircraft and satellite remote sensing platforms for precision viticulture. *Remote Sensing*, 7(3):2971–2990, 2015.
- [82] A. Milella, G. Reina, and J. Underwood. A self-learning framework for statistical ground classification using radar and monocular vision. *Journal of Field Robotics*, 32, 2015.
- [83] Annalisa Milella, Roberto Marani, Antonio Petitti, and Giulio Reina. In-field high throughput grapevine phenotyping with a consumer-grade depth camera. *Computers and Electronics in Agriculture*, 156:293–306, 2019.
- [84] Annalisa Milella and Giulio Reina. 3d reconstruction and classification of natural environments by an autonomous vehicle using multi-baseline stereo. *Intelligent Service Robotics*, 7(2):79–92, 2014.

- [85] Tharindu P Miyanawala and Rajeev K Jaiman. An efficient deep learning technique for the navier-stokes equations: Application to unsteady wake flow dynamics. *arXiv preprint arXiv:1710.09099*, 2017.
- [86] MG Harinarayanan Nampoothiri, B Vinayakumar, Youhan Sunny, and Rahul Antony. Recent developments in terrain identification, classification, parameter estimation for the navigation of autonomous robots. *SN Applied Sciences*, 3:1–14, 2021.
- [87] Francisco Yandun Narváez, Eduard Gregorio, Alexandre Escolà, Joan R Rosell-Polo, Miguel Torres-Torriti, and Fernando Auat Cheein. Terrain classification using tof sensors for the enhancement of agricultural machinery traversability. *Journal of Terramechanics*, 76:1–13, 2018.
- [88] NASA. Jpl. mars exploration rovers. <http://marsrovers.jpl.nasa.gov/home/index.html>. Accessed: 2023-2-28.
- [89] NASA. Nasa mars. <https://mars.nasa.gov/mars2020/>. Accessed: 2023-2-28.
- [90] James Nelson and Stefano Sanvito. Predicting the curie temperature of ferromagnets using machine learning. *Physical Review Materials*, 3(10):104405, 2019.
- [91] Jorge Ocón, Iulia Dragomir, Andrew Coles, A Green, L Kunze, R Marc, CJ Perez, T Germa, V Bissonnette, G Scalise, et al. Ade: Autonomous decision making in very long traverses. 2020.
- [92] Lauro Ojeda, Johann Borenstein, Gary Witus, and Robert Karlsen. Terrain characterization and classification with a mobile robot. *Journal of Field Robotics*, 23:103–122, 2006.
- [93] Lauro Ojeda, Giulio Reina, Daniel Cruz, and Johann Borenstein. The flexnav precision dead-reckoning system. *International Journal of Vehicle Autonomous Systems*, 4(2-4):173–195, 2006.
- [94] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.
- [95] Kyohei Otsu, Masahiro Ono, Thomas J Fuchs, Ian Baldwin, and Takashi Kubota. Autonomous terrain classification with co-and self-training approach. *IEEE Robotics and Automation Letters*, 1(2):814–819, 2016.
- [96] Sebastian Otte, Stefan Laible, Richard Hanten, Marcus Liwicki, and Andreas Zell. Robust visual terrain classification with recurrent neural networks. *Proceedings; Presses Universitaires de Louvain: Bruges, Belgium*, pages 451–456, 2015.

- [97] Chandramouli Padmanabhan, Sayan Gupta, Annadurai Mylswamy, et al. Estimation of terramechanics parameters of wheel-soil interaction model using particle filtering. *Journal of Terramechanics*, 79:79–95, 2018.
- [98] Panagiotis D Paraschos, Georgios K Koulinas, and Dimitrios E Koulouriotis. Reinforcement learning for combined production-maintenance and quality control of a manufacturing system with deterioration failures. *Journal of Manufacturing Systems*, 56:470–483, 2020.
- [99] Jongwon Park, Kyushik Min, Hayoung Kim, Woosung Lee, Gaehwan Cho, and Kunsu Huh. Road surface classification using a deep ensemble network with sensor feature selection. *Sensors (Basel)*, 18, 2018.
- [100] Ricardo Silva Peres, Jose Barata, Paulo Leitao, and Gisela Garcia. Multistage quality control using machine learning in the automotive industry. *IEEE Access*, 7:79908–79916, 2019.
- [101] Antonio Petitti, Fabio Vulpi, Roberto Marani, and Annalisa Milella. A self-calibration approach for multi-view RGB-D sensing. In Ettore Stella, editor, *Multi-modal Sensing and Artificial Intelligence: Technologies and Applications II*, volume 11785, pages 50 – 55. International Society for Optics and Photonics, SPIE, 2021.
- [102] Ashwin Rajkumar, Fabio Vulpi, Satish Reddy Bethi, Preeti Raghavan, and Vikram Kapila. Usability study of wearable inertial sensors for exergames (wise) for movement assessment and exercise. *Mhealth*, 7, 2021.
- [103] Ashwin Rajkumar, Fabio Vulpi, Satish Reddy Bethi, Hassam Khan Wazir, Preeti Raghavan, and Vikram Kapila. Wearable inertial sensors for range of motion assessment. *IEEE sensors journal*, 20(7):3777–3787, 2019.
- [104] Arianna Rana, Fabio Vulpi, Rocco Galati, Annalisa Milella, and Antonio Petitti. A pose estimation algorithm for agricultural mobile robots using an rgb-d camera. In *2022 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*, pages 1–5, 2022.
- [105] G. Reina and A. Milella. Toward autonomous agriculture: automatic ground detection using trinocular stereovision. *Sensors*, 12:12405–12423, 2012.
- [106] G Reina, A. Milella, R. Rouveure, M. Nielsen, R. Worst, and M. R. Blas. Ambient awareness for agricultural robotic vehicles. *Biosystems engineering*, 146:114–132, 2016.
- [107] Giulio Reina, Mario Foglia, Annalisa Milella, and Angelo Gentile. Rough-terrain traversability for a cylindrical shaped mobile robot. In *Proceedings of the IEEE International Conference on Mechatronics, 2004. ICM'04.*, pages 148–153. IEEE, 2004.

- [108] Giulio Reina, Antonio Leanza, and Giacomo Mantriota. Model-based observers for vehicle dynamics and tyre force prediction. *Vehicle System Dynamics*, 60(8):2845–2870, 2022.
- [109] Giulio Reina, Antonio Leanza, and Arcangelo Messina. Terrain estimation via vehicle vibration measurement and cubature kalman filtering. *Journal of Vibration and Control*, 26(11-12):885–898, 2020.
- [110] Giulio Reina, Antonio Leanza, Annalisa Milella, and Arcangelo Messina. Mind the ground: a power spectral density-based estimator for all-terrain rovers. *Measurement*, 151:107136, 2020.
- [111] Giulio Reina, Annalisa Milella, and Rocco Galati. Terrain assessment for precision agriculture using vehicle dynamic modelling. *Biosystems engineering*, 162:124–139, 2017.
- [112] Giulio Reina, Annalisa Milella, Raphaël Rouveure, Michael Nielsen, Rainer Worst, and Morten R. Blas. Ambient awareness for agricultural robotic vehicles. *Biosystems Engineering*, 146:114–132, 2016. Special Issue: Advances in Robotic Agriculture for Crops.
- [113] Giulio Reina, Lauro Ojeda, Annalisa Milella, and Johann Borenstein. Wheel slippage and sinkage detection for planetary rovers. *IEEE/Asme Transactions on Mechatronics*, 11(2):185–195, 2006.
- [114] Giulio Reina, Andres Vargas, Keiji Nagatani, and Kazuya Yoshida. Adaptive kalman filtering for gps-based mobile robot localization. In *2007 IEEE International Workshop on Safety, Security and Rescue Robotics*, pages 1–6, 2007.
- [115] Guy Revach, Nir Shlezinger, Ruud J. G. van Sloun, and Yonina C. Eldar. Kalmanet: Data-driven kalman filtering. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3905–3909, 2021.
- [116] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, M.I. Jordan, and S.S. Sastry. Kalman filtering with intermittent observations. *IEEE Transactions on Automatic Control*, 49(9):1453–1464, 2004.
- [117] David Stavens and Sebastian Thrun. A self-supervised terrain roughness estimator for off-road autonomous driving. *arXiv preprint arXiv:1206.6872*, 2012.
- [118] Matthias Stettler, Thomas Keller, Peter Weisskopf, Mathieu Lamandé, Poul Lassen, Per Schjønnning, et al. Terranimo®—a web-based tool for evaluating soil compaction. *Landtechnik*, 69(3):132–138, 2014.
- [119] Lei Tai, Shaohua Li, and Ming Liu. Autonomous exploration of mobile robots through deep neural networks. *International Journal of Advanced Robotic Systems*, 14(4):1729881417703571, 2017.

- [120] Angelo Ugenti, Fabio Vulpi, Raúl Domínguez, Florian Cordes, Annalisa Milella, and Giulio Reina. On the role of feature and signal selection for terrain learning in planetary exploration robots. *Journal of Field Robotics*, 39(4):355–370, 2022.
- [121] Angelo Ugenti, Fabio Vulpi, Annalisa Milella, and Giulio Reina. Learning and prediction of vehicle-terrain interaction from 3d vision. In *Multimodal Sensing and Artificial Intelligence: Technologies and Applications II*, volume 11785, pages 167–173. SPIE, 2021.
- [122] Inam Ullah, Xin Su, Jinxiu Zhu, Xuewu Zhang, Dongmin Choi, and Zhenguo Hou. Evaluation of localization by extended kalman filter, unscented kalman filter, and particle filter-based techniques. *Wireless Communications and Mobile Computing*, 2020, 2020.
- [123] Abhinav Valada and Wolfram Burgard. Deep spatiotemporal models for robust proprioceptive terrain classification. *The International Journal of Robotics Research*, 36:1521–1539, 2017.
- [124] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 1999.
- [125] F Vulpi, A Milella, F Cordes, R Dominguez, and G Reina. Deep terrain estimation for planetary rovers. In *15th International Symposium on Artificial Intelligence, Robotics and Automation in Space, ISAIRAS-2020*, 2020.
- [126] Fabio Vulpi, Antonio Leanza, Antonio Petitti, Annalisa Milella, and Giulio Reina. Kalman supervised network for improved model predictions. In *2022 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*, pages 1–7. IEEE, 2022.
- [127] Fabio Vulpi, Roberto Marani, Antonio Petitti, Giulio Reina, and Annalisa Milella. An rgb-d multi-view perspective for autonomous agricultural robots. *Computers and Electronics in Agriculture*, 202:107419, 2022.
- [128] Fabio Vulpi, Annalisa Milella, Roberto Marani, and Giulio Reina. Recurrent and convolutional neural networks for deep terrain classification by autonomous robots. *Journal of Terramechanics*, 96:119–131, 2021.
- [129] S Wang. *Road Terrain Classification Technology for Autonomous Vehicle*. Springer, Singapore, 2019.
- [130] C Weiss, H Frohlich, and A Zell. Vibration-based terrain classification using support vector machines. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Beijing, China, October 2006.
- [131] Christian Weiss, Matthias Stark, and Andreas Zell. Svms for vibration-based terrain classification. In *Autonome Mobile Systeme 2007*, pages 1–7. Springer, 2007.

- [132] Lorenz Wellhausen, Alexey Dosovitskiy, René Ranftl, Krzysztof Walas, Cesar Cadena, and Marco Hutter. Where should i walk? predicting terrain properties from images via self-supervised learning. *IEEE Robotics and Automation Letters*, 4(2):1509–1516, 2019.
- [133] Dong-Fan Xie, Zhe-Zhe Fang, Bin Jia, and Zhengbing He. A data-driven lane-changing model based on deep learning. *Transportation research part C: emerging technologies*, 106:41–60, 2019.
- [134] Nan Xu, Zepeng Tang, Hassan Askari, Jianfeng Zhou, and Amir Khajepour. Direct tire slip ratio estimation using intelligent tire system and machine learning algorithms. *Mechanical Systems and Signal Processing*, 175:109085, 2022.
- [135] Wei Xu, Yixi Cai, Dongjiao He, Jiarong Lin, and Fu Zhang. Fast-lio2: Fast direct lidar-inertial odometry. *IEEE Transactions on Robotics*, 2022.
- [136] Liang Yu, Shuqi Qin, Meng Zhang, Chao Shen, Tao Jiang, and Xiaohong Guan. A review of deep reinforcement learning for smart building energy management. *IEEE Internet of Things Journal*, 8(15):12046–12063, 2021.
- [137] Hong Zhang, Hoang Nguyen, Xuan-Nam Bui, Biswajeet Pradhan, Panagiotis G Asteris, Romulus Costache, and Jagannath Aryal. A generalized artificial intelligence model for estimating the friction angle of clays in evaluating slope stability using a deep neural network and harris hawks optimization algorithm. *Engineering with Computers*, pages 1–14, 2021.
- [138] Kai Zhao, Mingming Dong, and Liang Gu. A new terrain classification framework using proprioceptive sensors for mobile robots. *Hindawi*, 2017:14, 2017.
- [139] Qinpei Zhao and Pasi Fränti. Wb-index: A sum-of-squares based index for cluster validity. *Data & Knowledge Engineering*, 92:77–89, 2014.

GATHERreferences.bib