Journal Article

Adaptive rover behaviour based on online empirical evaluation
Rover-terrain interaction and near to far learning

Author(s):
Krebs, Ambroise; Pradalier, Cédric; Sigwart, Roland

Publication Date:
2010

Permanent Link:
https://doi.org/10.3929/ethz-a-010001815

Rights / License:
In Copyright - Non-Commercial Use Permitted
Abstract

Due to the fundamental nature of all-terrain exploration, autonomous rovers are confronted with unknown environments. This is especially apparent regarding to soil interactions, as the nature of the soil is typically unknown. This work aims at establishing a framework from which the rover can learn from its interaction with the terrains encountered and shows the importance of such a method. We introduce a set of rover-terrain interaction (RTI) and remote data metrics which are expressed in different subspaces. In practice, the information characterizing the terrains, obtained from remote sensors (e.g. a camera) and local sensors (e.g. an Inertial Measurement Unit) is used to characterize the respective remote data and rover-terrain interaction (RTI) model. In each subspace, which can be described as a feature space encompassing either a remote data measurement or a RTI, similar features are grouped to form classes, and the probability distribution function over the features is learned for each one of those classes. Subsequently, data acquired on the same terrain is used to associate the corresponding models in each subspace and to build an inference model. Based on the remote sensor data measured, the
RTI model is predicted using the inference model. This process corresponds to a near to far approach and provides the most probable RTI metrics of the terrain lying ahead of the rover. The predicted RTI metrics are then used to plan an optimal path with respect to the RTI model and therefore influence the rover trajectory. The CRAB rover is used in this work for the implementation and testing of the approach, which we call Rover Terrain Interactions Learned from Experiments (RTILE). This article presents RTILE, describes its implementation and concludes with results from field tests that validate the approach.
1 Introduction

The field of mobile robotics has attracted a lot of attention in recent years, which is also due to very famous missions such as planetary exploration with the Mars Exploration Rover (MER), the Mars Science Lab (MSL), or ExoMars, and competitions such as the DARPA Grand Challenge or AUVSI. Although robotic platforms have become very popular, they are only a means to perform a more valuable task and not really an achievement themselves. Therefore, and especially in the context of all-terrain robotic platforms, robotic rovers have to be designed as best as possible with respect to their task (what has to be done), their context (where it has to be done) and their mission constraints (how it has to be done). For example, in the context of planetary exploration, the robot is required to provide a safe and reliable locomotion with optimal performance, while consuming as little energy as possible, in order to accomplish scientific analysis in the area of interest.

In the end, the performance of the robot is influenced by two factors:

- First, the performance depends on the robot’s physical and mechanical properties, corresponding to its structure, suspension mechanism, actuators and sensors.
- Second, the performance is also related to the control of the robotic platform, in a very generic sense. This includes research in fields such as control, obstacle avoidance, path planning, pose estimation, and so forth.

As mentioned in the latter point, the robot’s performance is related to the interaction of the robot with its surroundings and its capability to sense and represent the environment. A natural environment, which is usually the operating place for all-terrain rovers, involves a great diversity in terrain, soils and obstacle types, shapes and appearances. This diversity is difficult to model and hence implies additional uncertainty that the rover must cope with.

1.1 Objectives

On the 26th of April 2005 (SOL 446\(^{1}\)), one of NASA’s two Martian rovers, Opportunity, almost got stuck in a sand dune in Meridiani Planum. It took five weeks for the scientists to cautiously extract the rover from this delicate situation and allow the mission to continue its course. This example shows the importance of the uncertainties that automatically come with an exploration mission. Although MER rovers were extremely well designed, and even if their interactions with expected terrains were cautiously characterized and tested in many situations, such unexpected events can still occur. In the end, the terrain on which the rover has to operate is, at least partially, unknown and is extremely difficult to characterize beforehand. This effect is even more apparent in the case of applications in rough-terrain. Hence we are interested to design a rover with the capability to learn from its experience while it operates in a mission. Then the accumulated knowledge could be used to improve the rover’s behaviour. Thus the idea is to make the best out of an operating platform by giving it the capability to link remote data and local data. Remote data describes the environment at a distance and is provided by sensors such as camera, LIDARs, and so forth. Local data

\(^{1}\)http://marsrovers.nasa.gov/mission/status_opportunityAll_2005.html#sol446
expresses the rover-terrain interaction (RTI) model which refers to a metric characterizing some aspect of the rover behaviour. The RTI model can be related to sensors such as an Inertial Measurement Unit (i.e. IMU), actuator encoders and other proprioceptive sensors. Learning the correspondence between local data (information that is near) and remote data (information that is far) allows anticipating the rover-terrain interaction characteristics ahead of robot position. This information can be used to influence the rover behaviour by changing its path. The data association can be referred to as *near to far* and is a concept at the very center of our approach. In fact, this data association allows, in our case, the inference of local data based on remote data, which corresponds to generating a RTI model of the terrain lying ahead of the rover.

### 1.2 State of the Art

In all-terrain robotics, one of the main tasks of a rover is to successfully move from a point $A$ (starting position), to a point $B$ (goal). The first aspect is to ensure that the rover uses only traversable areas to reach the goal. This criterion is fundamental but it usually does not limit the possible paths to a single trajectory. The different possible paths can be more or less efficient according to the RTI model. The research of the present article proposes an approach allowing the RTI model, and not the traversability, to be asserted based on remote data. Based on this knowledge, the path with the best RTI can be used. This subsection describes the main research trends related to this work as trying to identify the terrain around the robot is not a new preoccupation in the field. They are presented along three topic lines. We begin by presenting the researches using *near to far* approaches enabling the traversability prediction. Then we describe the approaches allowing recognizing or identifying the terrain on which the rover is driving, making use of an RTI model. Finally, the approaches enabling the RTI predictions are described. The last part relates to the present work, and its specificities with respect to other works described in this state of the art.

There are several research projects trying to assert whether the terrain ahead is traversable or not, before negotiating it. These remote based methods use different combinations or types of remote sensors – including radars, vision cameras, and laser range finders – e.g. (Lalonde et al., 2006) (Manduchi et al., 2005) (Poppinga et al., 2008) (Vandapel et al., 2004). The asserted traversability is then used to plan the robot’s route towards the goal and to ensure its safety. Since 2005, the DARPA-funded LAGR$^2$ project (Mulligan and Grudic, 2006b), (Mulligan and Grudic, 2006a) contributed enormously to the development and integration of these approaches. It offered a common robotic platform to develop and test off-road algorithms and compare the results. Trials were regularly organised, offering challenging environment in which the robot had to find its way to reach the goal, and learn from its experiments. Thus, the different teams involved in the project, such as (Bajracharya et al., 2009) (Huang et al., 2009) (Kim et al., 2006) (Konolige et al., 2009) (Otte et al., 2009) (Procopio et al., 2009) (Sermanet et al., 2009) showed solution to enable a robot to learn from its experiment and handle unexpected events. Nevertheless, these researches are focused on giving an answer to whether the terrain is traversable or not. This is expected as it is in line with the goals of the LAGR project which offers the rover several trials to reach a goal.

$^2$LAGR stands for Learning Applied to Ground Robots
enabling the rover to learn from its previous experiences. Furthermore, (Hadsell et al., 2009) and (Happold et al., 2006) use an image-based classification to enhance the traversability estimation of the terrain. Thus, from near to far, the approach becomes further to near, extending the perception horizon of the rover. In (Wellington and Stentz, 2004), the true ground height in vegetation is computed by making use of a near to far approach. Again this information allows the autonomous vehicle to define whether the terrain is traversable or not.

In parallel, a line of research proposes to classify terrains based on the rover-terrain characteristics (Ojeda et al., 2006), such as the vibrations induced in the robot (Iagnemma et al., 2004). Vibration is mainly measured in terms of the linear acceleration of the wheel-bars or the robot body (Brooks and Iagnemma, 2005) (Brooks and Iagnemma, 2007) (Weiss et al., 2006). See (Weiss et al., 2007) for a comparison of different methods for classification of three and six types of terrains using a frequency-based representation. In relation with the vibration characteristics, the effect of the robot’s velocity on the terrain classification is studied in (Dupont et al., 2008) and a velocity independent classification method is proposed and tested on a very limited number of terrains. Similarly, (Ward and Iagnemma, 2009) proposes a speed independent solution for vehicles with a higher dynamics, classifying terrain using a single suspension mounted accelerometer. Another famous characteristic used to express the rover-terrain behaviours is slippage. (Angelova et al., 2007) proposes to learn the slippage model of the rover with respect to the terrain type and geometrical characteristics. (Ishigami et al., 2006) also proposes a slippage model in relation with the terrain slope. In (Weiss and Zell, 2008) it is argued that an autonomous system, in addition to learning from training data, should be able to detect and classify new terrains. The authors propose a Gaussian mixture model for detection and classification of novel terrains. (Halatci et al., 2008) proposes a classification method making use of both visual and vibration data, comparing several classification methods. It shows great results at classifying terrains encountered but fundamentally the process is a supervised one.

Apart from (Angelova et al., 2007) and (Brooks and Iagnemma, 2007), there is no near to far approach used to anticipate the rover-terrain interaction. In (Angelova et al., 2007), the terrain appearance is divided into well known classes providing a prior set of trained classifiers. The number and type of the terrain is then fixed and the slippage model is then learned for each one of those. The end-to-end approach including this work is presented in (Helmick et al., 2009). In the case of (Brooks and Iagnemma, 2007), it is the other way around as the visual characteristics of the terrain are learned online while the rover-terrain characteristics classifiers were trained before hand. One way or another, those approaches rely on trained classifiers which implies a given and fix number of classes.

1.3 Contributions

The research of the present article proposes an approach allowing the RTI model, and not the traversability (as in the LAGR-related projects), to be asserted based on remote data. Thus, based on a near to far approach, the rover terrain interactions can be estimated and abstracted into classes. We also argue that it seems more appropriate for a robot to classify terrains based on its needs and according to how they affect its behaviour and not necessarily
based on human defined classes. In other words, the presented approach categorizes the terrain in a way that is suitable for its path planning. Finally, another essential point is that this classification is not fixed to a rigid number of classes, but rather resulting from the rover’s experience. The terrain representation is actually learned and can evolve online with respect to the situation encountered by the rover. Therefore, the approach is very flexible and is capable of incorporating new types of terrain or RTI models while traversing the terrain. Also note that the present work does not take the traversability into account since the focus lies instead on the RTI model learning and prediction. The overall approach is named RTILE, which stands for Rover Terrain Interactions Learned from Experiments.

1.4 Content

The objective of our work is to implement an online terrain classification and prediction algorithm. Its goal is to modify the rover’s behaviour according to a given metric. This metric could be, for example, the slippage minimization in the context of the MER rover. The next section focuses on giving an overview of the approach, including a theoretical description of all the components needed to achieve it. Then, the implementation of this approach in the specific context of the CRAB rover platform is described in Section 3. The CRAB is a rover with six motorized wheels (Fig. 1). We present the results of three described experiments in section 4. The first one shows the learning capabilities of the approach while the other two expose the overall effects of the approach, namely, the rover’s behaviour being influenced by its experience. The last part of the article is dedicated to the conclusion and future work envisioned.
2 Approach Overview

This section describes our approach to tackle the rover-terrain interaction problem. The focus is placed on the theoretical and generic concepts of RTILE. Fig. 2 shows a simplified description of the approach and highlights its most important elements. First the near to far part is realized using a grid-based approach, allowing for local and remote data associations. A Bayesian model (Bessière et al., 2008) is used to handle the uncertainties and its various probability distributions are asserted in the learning part. The prediction makes use of the knowledge acquired to estimate the RTI ahead of the rover. Finally the path planning uses the predicted RTI to influence the rover trajectory.

2.1 Rover Behaviour

The rover behaviour refers to its manner of behaving or acting within its environment. It can be influenced in many different ways as it is dependant on various levels of control. In many research projects, the elements acted upon are the control type or the control parameters; however, in this study, we are interested in the path used by the rover. According to that, the traversability, in other words where the robot is able to drive or not, is essential to the robot. The focus is on a complementary aspect, namely to differentiate the areas traversable with respect to their trafficability – ability of a rover to traverse different soil type – from their terrainability – ability to negotiate terrain irregularities – (Apostolopoulos, 2001). Specific metrics are used here to learn and characterize the rover’s performance. For example, in the case of the rover Opportunity, as well as other rough-terrain robots, a possibility would be
to use a metric referring to the slippage. However our approach is not limited to slippage. The metrics used in the context of this research are presented in the next section.

As an example is the robot Roomba\textsuperscript{3}, a very famous domestic robot aimed at cleaning home floors. The robot behaves the same regardless of the type of floor (parquet, carpet, etc.) of the diverse areas it has to clean. It would be nice to make this robot infer the initially unknown, diverse types of floors from the noise the rover makes while cleaning them. Thus the Roomba could optimize its cleaning pattern according to a metric driving the rover behaviour, which in this case could be the amount of noise generated. For example, the robot could prefer to clean the area where it makes less noise at night.

2.2 Near to Far

The capability of a rover to learn from its environment is linked with its ability to acquire data from the environment; which is directly linked with the rover sensors. Two different types of sensors are needed here; namely local and remote. First, we require the ability to characterize aspects of the rover terrain interactions with the local sensors. Then, in order to influence the rover’s behaviour, i.e. its path, we need to obtain information of the terrain ahead of the vehicle with the remote sensors. The remote sensors observe the terrain ahead of the vehicle and extract some terrain characterizations. Later on, when driving over the same terrain patch, the behavioural sensor records another terrain characterization. It is important to link both these data, to generate a bridge from near to far. The remote characteristics (or features) are expressed as $F_r$ and $F_l$ corresponds to the local ones. The goal then is to form samples $s_i$ associating the corresponding remote and local features, samples that can be then processed and used for learning.

$$s_i = (F_l, F_r)$$

(1)

It is important to note that in eq. 1 the remote and local features correspond to the same spatial area in order to have consistent $s_i$. Thus, the $F_l$ and $F_r$ are not acquired at the same time and the use of $F_r$ is delayed to form the samples.

2.3 Probabilistic Model

Based on the samples reflecting the rover experience, a predictive model can be built. This model is used to associate predictive terrain observations and behavioural terrain characterizations (roughness, slippage, softness, etc.). Therefore, the joint probability distribution and its decomposition includes $F_r$ and $F_l$.

$$P(F_r, F_l) = P(F_r)P(F_l | F_r)$$

Such a model, although theoretically correct, leads to complexity and dimensionality issues.

\textsuperscript{3}http://www.irobot.com
In fact, the computed features are continuous objects in multiple dimensions and the corresponding distributions are difficult to implement and compute. For this reason similar features are regrouped into classes, providing an abstraction layer. Thus $K_r$ and $K_l$, which express the terrain type (or class) in both the remote and local feature space, are added to the model.

$$P(F_r, F_l, K_r, K_l).$$

The decomposed joint distribution is expressed as follow:

$$P(F_r, F_l, K_r, K_l) = P(K_r)P(K_l | K_r)P(F_l | K_l)P(F_r | K_r).$$ \hfill (2)

The probability distribution of the features is assumed to be dependant only on its corresponding class. This means the features expressing different RTIs, $F_l$, and remote terrain information, $F_r$, are conditionally independent, given $K_l$ and $K_r$. This is an important assumption that drives the RTI model and the corresponding features design. In the end, the idea is to use a more tractable expression to the following equation:

$$P(F_l | F_r)$$ \hfill (3)

which corresponds to the following question:
What are the predicted $F_l$, based on the observed $F_r$?

### 2.4 Inference

Now that the probabilistic model has been expressed and its joint distribution described, the Bayes’ rule can be used to develop eq. 3.

$$P(F_l | F_r) = \frac{\sum_{K_r} \sum_{K_l} P(K_r)P(K_l | K_r)P(F_l | K_l)P(F_r | K_r)}{P(F_r)}$$

$$= \sum_{K_r} \left[ \left( \frac{P(K_r)P(F_r | K_r)}{P(F_r)} \right) \sum_{K_l} P(F_l | K_l)P(K_l | K_r) \right] \hfill (4)$$

In order to simplify this expression, we can remark that the first part of this equation corresponds to the development of $P(K_r | F_r)$ using Bayes’ rule:

$$P(K_r | F_r) = \frac{P(K_r)P(F_r | K_r)}{P(F_r)}$$ \hfill (5)
Also, using the law of total probability, we can identify the second part of eq. 4 as $P(F_l | K_r)$, the distribution over the local features knowing the remote class:

$$P(F_l | K_r) = \sum_{K_l} P(F_l | K_l) P(K_l | K_r)$$

$$= \sum_{K_l} P(F_l | K_l) P(K_l | K_r)$$  \hspace{1cm} (6)

Note that this uses the assumption that the local features are conditionally independent of the remote class knowing the local class: $P(F_l | K_l K_r) = P(F_l | K_l)$.

Using eq. 5 and eq. 6, eq. 4 can be rewritten as:

$$P(F_l | F_r) = \sum_{K_r} \left[ P(K_r | F_r) \sum_{K_l} P(F_l | K_l) P(K_l | K_r) \right]$$

$$= \sum_{K_r} P(K_r | F_r) P(F_l | K_r)$$  \hspace{1cm} (7)

To simplify the double summation over $K_r$ and $K_l$, we will only consider the most likely remote class based on the remote features. Let us define the most likely remote class $\tilde{k}_r$ as:

$$\tilde{k}_r = \arg\max_{K_r} (P(K_r | F_r))$$  \hspace{1cm} (8)

Only considering $\tilde{k}_r$ is translated into the following approximation:

$$P(K_r = \tilde{k}_r | F_r) \approx 1 \quad \text{and} \quad P(K_r \neq \tilde{k}_r | F_r) \approx 0,$$  \hspace{1cm} (9)

which is a good approximation, as long as the classes are well separated in the features space, meaning that their probability distributions have to be peaked.

Using this approximation, eq. 7 becomes:

$$P(F_l | F_r) \approx P(K_r = \tilde{k}_r | F_r) \sum_{K_l} P(F_l | K_l) P(K_l | K_r = \tilde{k}_r)$$

$$\approx \sum_{K_l} P(F_l | K_l) P(K_l | K_r = \tilde{k}_r)$$

$$\approx P(F_l | K_r = \tilde{k}_r)$$  \hspace{1cm} (10)

Therefore eq. 7 and 10 allow us to determine the most probable predicted local features $F_l$ based on the remote features $F_r$.

Note that summing all the probabilities in the computation of $P(F_l | K_r)$ is the mathematically exact way to proceed, but approximations could be considered, resulting in a pessimistic or optimistic result. For example a pessimistic approach would only consider the local feature $F_l$ resulting in the worst robot behaviour, among the ones that can be predicted from the remote feature class $K_r$. Of course, the meaning of worst has to be defined in the RTI context and can be asserted by the impact of the $F_l$ on the global rover performance. Such an approach is an interesting way of preserving the hardware. In any case, asserting that the $F_l$ is predicted from the $F_r$, corresponds to predicting the RTI.
The theoretical description of the approach is expressed involving several probability distribution functions, which are learned based on the rover experience. The learning process is the subject of the two next subsections.

### 2.5 Class Association

In this subsection, we expose the connection between the probability distribution of the local classes and the remote ones, which corresponds to expressing the link between the classes on the local and remote space, or $P(K_l | K_r)$ from eq. 2. This information is acquired at the same time as the samples $s_i$ are formed, since it contains the correspondence between the data in each subspace, as shown in Fig. 3.

![Figure 3: Connections between the classes of different subspaces allowing inference.](image)

The samples, $s_i$, provide knowledge about how probable the connections between the various subspaces are. Thus assuming that $A$ and $B$, in Fig. 3, are two subspaces with respectively two and three classes,

$$P(K_B = 3 | K_A = 2) = \frac{N_{K_B=3K_A=2}}{N_{K_A=2}} = \frac{P(AB)}{P(A)}$$

With $N_{K_B=3K_A=2}$ being the number of connections between class three of $B$ and class two of $A$ and $N_{K_A=2}$ representing the number of samples collected as part of class two of $A$. Now assuming that $A$ is a remote subspace and $B$ is a local subspace, the class association can be performed and it is learned based on the data obtained by the rover.

### 2.6 Class Definition & Novelty Detection

Besides learning the class association, the ability to learn the classes themselves within the features space is fundamental. In other words, we need a method to decide whether a new data is part of an existing class, or if it is eligible for creating a new one. An important aspect of the learning is to let the robot handle the data related to its experiments in the most suitable way. This means that the classes are not necessarily corresponding to human defined criteria. In this sense, the approach relates closely to unsupervised learning.
The probability distributions of the features for a given class are defined as a Gaussian distribution. The inference and the expression of the prediction are described in the following part.

Assume we have a feature space consisting of \( C \) classes, which distributions are Gaussian, to which another distribution is added. This additional distribution refers to the unknown class, or class 0 and is based on a uniform distribution. Therefore, the probability distributions are the following:

\[
P(F \mid K = k) = \begin{cases} 
G_{\mu_k, \sigma_k} & \text{if } 1 \leq k \leq C \\
U & \text{otherwise } (k = 0)
\end{cases}
\]  

With \( F \) and \( K \) being variables associated respectively to the features and the class number. The novelty detection aims to know the class corresponding to a given feature. If the class is not known yet, a new class can be created. Therefore, the question is the following:

\[
P(K \mid F) \propto P(K)P(F \mid K).
\]

And using the maximum likelihood, it can be expressed as:

\[
\hat{k} = \arg\max_{j \in [1, n]} (P(F \mid K = j)).
\]

Knowing that, the novelty detection concerns data whose classification result in \( K = 0 \). For example, considering a subspace of one dimension with a \( j \) classes learned, a new data \( F = f \) is classified as follow:

\[
K = \begin{cases} 
\hat{k} & \text{if } P(K = \hat{k} \mid F = f) > U \\
0 & \text{otherwise}
\end{cases}
\]

In case that the classification results in \( K = 0 \) the new data is considered as not being part of the existing class and instead is considered to create a new one. Most importantly, this decision is taken automatically, naturally when the new data does not fit into the current description of the classes in the subspace.

The data are processed in batches, at a specific moment (e.g. when the rover has traveled a given distance, or reached a waypoint). At this point, the new pieces of data available are handled as follow:

- If it can be found as part of an existing class, it will then reinforce the knowledge of this class.
- If it can be found as part of class 0, or the unknown class, then the new data is part of no existing class in the current representation. Thus we have the following possibilities:
– The new data \( f \) can be used, together with \( N \) other similar ones, to create a new class \( K = C + 1 \).

\[
P(F \mid K = C + 1) = G_{\mu_{C+1}, \sigma_{C+1}}
\]

with

\[
\mu_{C+1} = \frac{1}{N} \sum_{i=1}^{N} f_i \quad \text{and} \quad \sigma^2_{C+1} = \frac{1}{N} \sum_{i=1}^{N} (f_i - \mu_{C+1})^2.
\]

– If perceived as an outlier, the data can be either stored for later use (when enough similar data are available), or simply discarded (in case of a transition between terrains for example). A new data can be perceived as an outlier if the total number of samples of the unknown class is not big enough.

### 2.7 Subspaces

The feature space representation, in which the learning is performed, is divided into a set of subspaces. Each one of these is a feature space representing either remote data or an RTI (i.e. local data). This implementation is performed for the following reasons:

Firstly, using all the inputs of the robots sensors to characterize its interaction with the terrain into a single space leads to implementations issues. The number of dimensions of the features space would be enormous and it would be ineffective since it would most likely be sparse. Furthermore, the implementation would be difficult due to computational costs. Thus, in order to have useful results and distributions with a reasonable amount of dimensions, a framework has to be defined.

Secondly, we are more interested by the meaning of the rover’s interaction with the terrain rather than the rover’s sensor data themselves. Therefore, the feature space is subdivided into subspaces that correspond to given characteristic of the RTI. The features of the subspaces themselves correspond directly to the characteristics, and they are learned with the assumption that they are distributed according to a (multi) normal law. For each subspace, the metric that corresponds to a good or bad RTI, is also known.

Finally, another advantage of subdividing the space is the flexibility it offers. The different RTI models can be treated one after the other and are independent. Thus adding a new type of RTI model, i.e. as a result of a new sensor integrated to the platform, does not affect what is already learned nor the other subspaces. This is particularly useful in the context of this research.

We propose to define the subspaces according to the terrain characteristics (e.g. softness) and their effects on the robot. It can be interesting to have only a single type of sensor per subspace even though several sensors provide data regarding the same RTI model. In this case, several subspaces addressing the same RTI model can be used. Such a methodology improves the approach’s reliability. Thus, if a sensor is damaged in operation, or if a sensor is modified or changed, part of the terrain representation can still be used.

The features of the subspaces, modeling the RTI, have to be designed with care. They are the elements that will capture the terrain representation and the class definitions within the
subspaces.

For example, in the context of the Roomba robot, the RTI type could be defined as the noise generated by the robot cleaning the soil. The feature that could be used in this case is a root mean square (RMS) value of the noise signal. In this case a low RMS value would correspond to a better RTI, if a quiet behaviour of the robot is preferred. To summarize, the subspaces have to be defined with respect to the robotic platform (the sensors available) and the application to which the robot is dedicated.

### 2.8 Path Planning

A path planner is used to drive the rover and reach the goal, using the terrains with the best predicted RTI model possible. This path planner is $E^*$ (Philippsen et al., 2006). It is briefly described here in a first part and then its use within RTILE is explained.

The $E^*$ algorithm is a grid-based path planner which is based on a weighted region approach. The environment in which the rover is evolving is represented as a grid where the robot position and its goal position are known. A navigation function is computed for each cell, stating the path’s cost to reach the goal from this cell. The underlying technique is expressed within the continuous domain, which corresponds to a wavefront propagating from the goal toward the rover. The path to reach the goal can be found by using a gradient descent over the navigation function.

The grid used by $E^*$, named $G_e$, is formed of nodes, or cells $c_e$. Several interesting properties are linked with $c_e$:

- $r(c_e)$ is the difficulty or cost of traversing a given cell. This parameter corresponds intuitively to the wavefront speed of the navigation function. For example, a cell corresponding to an obstacle would block the wavefront.
- $v(c_e)$ which represents the “height” of the navigation function of the cell. It can be computed based on the $v(c''_e)$ of the neighbor cell closest to the goal, $c''_e$, and based on the wavefront propagation cost $r(c_e)$.

In a typical application, the rover and goal positions are given to $E^*$. The navigation function is then computed for each $c_e$, based on the $r(c_e)$. The cells corresponding to obstacles block the navigation function propagation and the cells in the neighborhood of those have their propagation cost increased. This pushes the rover away from the obstacles without blocking the navigation function propagation. The trajectory reaching the goal can be then computed by using the gradient descent on the navigation function, from the rover position, to the goal position.

In our work the propagation cost is used in a different way. It is computed based on traversability, as well as on the predicted RTI. Thus, we have:

---

4$E^*$ is very similar to the field D* (Ferguson and Stentz, 2005) regarding the functionalities. It differs mostly in terms of implementations.
\[ r(c_e) = f(\Lambda, T). \] (13)

The metric \( \Lambda \) depends on the evaluation of the RTI, called \( M_{RTI} \), computed based on the local features. \( T \) depends mainly on the geometry of the terrain and can be considered as a Boolean value corresponding to the traversability. Even though other research works consider the traversability as a continuous metric, a decision has still to be taken allowing the rover to traverse or not an area. \( T \) is the result of this decision.

\[ \Lambda = h(M_{RTI}) \text{ with } \Lambda \in [0; 1] \] (14)

\[ T = \begin{cases} 
1 & \text{if the cell is traversable} \\
0 & \text{otherwise}
\end{cases} \] (15)

\( \Lambda \) takes a high value for a terrain having a good rover interaction. Hence, a value of 1 for \( \Lambda \) means a perfect behavior of the rover according to the metric defined, while a value of 0 corresponds to a terrain to be avoided. For example, assuming that the rover faces two terrains (named white and gray), as depicted in Fig. 4. The rover, which position is marked by the cross, has to reach the goal on the right hand side. If the rover is able to identify the terrains according to:

\[ \Lambda_{\text{white}} > \Lambda_{\text{gray}} \]

then the resulting generated trace (green dashed) provided by \( E^* \) naturally avoids the gray terrain and is a result of a gradient descent performed on the navigation function (illustrated by the wavefront in blue). In summary, \( E^* \) offers a trade off between the movement cost and the path length and it provides a trace to be followed to reach the goal.

Figure 4: Wavefront propagation with \( E^* \) (Philippsen, 2006), from the goal (marked with a circle) to the robot (marked with a cross). The trace proposed by \( E^* \), using the gradient descent is also depicted in dashes.
3 Implementation

In this section, the RTILE approach is described in more details as well as the challenges and solutions implemented in the context of the CRAB rover platform. The most important assumptions are first described, and the rover is then presented. The learning process and its use to predict the RTI follows. Finally, the pseudo code of the approach and a summary conclude this section.

3.1 Assumptions

The most important assumptions driving the implementation of the approach on the CRAB rover, and therefore influencing the interpretation of the tests results, are summarized as follows:

- While numerous works are identifying the traversability \( T \) of the rover surrounding, the goal here focuses on the rover-terrain interaction. The consequence is that the traversability is not taken into account, or rather, the terrain surrounding the rover is assumed traversable.
- The terrain surrounding the rover is assumed to be globally flat. As we are interested in showing the adaptive behaviour of the robot, the experiments conducted uses flat grounds with varied \( \Lambda \) characteristics.
- Knowledge acquired during past tests can be reused, but this is not necessarily. It is assumed that the approach is not dependent on any prior knowledge regarding the RTI or the remote data model. Thus, a predefined set of classes, such as grass, gravel and so on, is not required.
- The adaptive behaviour is driven by \( \Lambda \) (eq. 14) which is assumed to be known and provided by a user. In other words, the meaning of what a ”good” and ”bad” terrain is must be provided.

3.2 Hardware Platform

The platform used in the context of this project is the CRAB rover, depicted on Fig. 1 and 5. The CRAB rover has six motorized wheels with a passive suspension mechanism. It is composed of a double parallel bogie mechanism on each side, connected via a differential. The first parallel bogie is between the front and the middle wheel while the second link the middle and the back wheel. The loop is closed via a rocker bogie which is connected to the chassis. As this link is made via a single pivot joint, a differential is necessary to control the chassis attitude. The kinematics of the passive suspension mechanism can be observed in Fig. 5. The pivot joints on the parallel bogie are positioned so that the rover has an equal repartition of its mass on the wheels on a level ground. The six wheels are motorized with DC Maxon motors, as well as the four steering units which are linked to the four corner wheels. The control of the robot can be performed according to the kinematic constraints by implementing a double Ackermann-steering, one for the front wheels and the other for the back wheels. Table 1 gives an overview of the most important CRAB dimensions.
Table 1: Dimensions of the CRAB rover.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Width</td>
<td>0.773</td>
<td>m</td>
</tr>
<tr>
<td>Distance front to middle</td>
<td>0.324</td>
<td>m</td>
</tr>
<tr>
<td>Distance middle to back</td>
<td>0.324</td>
<td>m</td>
</tr>
<tr>
<td>Ground clearance</td>
<td>0.2</td>
<td>m</td>
</tr>
<tr>
<td>Wheel diameter</td>
<td>0.196</td>
<td>m</td>
</tr>
<tr>
<td>Wheel width</td>
<td>0.1</td>
<td>m</td>
</tr>
<tr>
<td>Weight</td>
<td>37.25</td>
<td>kg</td>
</tr>
<tr>
<td>Speed autonomous</td>
<td>0.06</td>
<td>$m \cdot s^{-1}$</td>
</tr>
</tbody>
</table>

The following sensors are available on the platform:

- An IMU, placed at the chassis level (represented by the green arrows in Fig. 5). It is a MT9-B from Xsens which provides Euler angles.

- Four angular sensors positioned on the parallel bogies. The sensors, $MR_b$, are home made, and based on a magneto resistive technology to measure the pivot joint angle at positions shown in Fig. 5 with $b$.

- Two angular sensors, $MR_d$, placed on the differential are of the same sensors as above. Their positions are shown in Fig. 5 with $d$.

- An HD webcam from Logitech (Quickcam Pro 9000) provides two mega pixel images of what lies ahead of the robot within a 43° field of view, and is the only sensor which provides predictive data.

![Figure 5: CRAB suspension mechanism movement. The position of the angular sensors is depicted with the red circles. The four $MR_b$ (two on each side) are indicated with a $b$ while the two $MR_d$ (one on each side) are marked with a $d$.](image-url)
3.3 Rover Control

The algorithm described below is used on the CRAB to follow the trace, or planned path, provided by E*. In a first step, the rover is considered as a differential-drive rover (referred to as differential rover). Then the computed commands are applied to the CRAB rover, taking its specific kinematic constraints into account.

![Figure 6: Path following for a differential-drive robot.](image)

The control for a differential rover can be achieved according to (Siegwart and Nourbakhsh, 2004). The translational ($v_{\text{trans}}$) and the rotational ($v_{\text{rot}}$) velocities are computed as follow:

$$v_{\text{trans}} = \begin{cases} k_p \rho & \text{if } \rho < \delta_1 \\ v_{\text{max}} & \text{otherwise} \end{cases}$$  \hspace{1cm} (16)

$$v_{\text{rot}} = k_\alpha \alpha$$  \hspace{1cm} (17)

With, $\rho$ the distance to the goal and $\alpha$ the angle between the rover’s orientation and the direct trajectory to the next trace waypoint, $t_r$. These parameters are illustrated on fig. 6.

$$t_r = \begin{cases} t_{r_i} & \text{if } \rho_w > \delta_2 \\ t_{r_i+1} & \text{otherwise} \end{cases}$$  \hspace{1cm} (18)

$t_r$ is defined as the next waypoint, $t_{r_i}$ except if it is closer than a threshold, $\delta_2$. In this case, the next waypoint is used. Thus, the rover reaches its goal following the trace provided by E*. The method described above is a standard approach which enables a differential robot to follow a trace. Table 2 shows the values of the control parameters used to drive the CRAB rover during the tests.

In order to retrieve the commands for each one of the ten motors (six driving wheels and four steering units), $v_{\text{trans}}$ and $v_{\text{rot}}$ have to be transformed into commands for the CRAB’s motors. The rover is basically controlled via a virtual wheel placed at the front of the rover, as depicted in Fig. 7. The virtual wheel’s steering angle $\eta_v$ and velocity $\omega_v$ are computed as follow:
Table 2: Control parameters for the CRAB rover.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k_\alpha$</td>
<td>0.2</td>
<td>$s^{-1}$</td>
</tr>
<tr>
<td>$k_\rho$</td>
<td>0.097</td>
<td>$s^{-1}$</td>
</tr>
<tr>
<td>$\delta_1$</td>
<td>0.285</td>
<td>$m$</td>
</tr>
<tr>
<td>$\delta_2$</td>
<td>0.507</td>
<td>$m$</td>
</tr>
</tbody>
</table>

\[
\eta_v = \arctan\left(\frac{v_{rot} \times d_l/2}{v_{trans}}\right) \quad (19)
\]

\[
\omega_v = \frac{v_{trans}}{\cos(\eta_v)} \quad (20)
\]

An instantaneous rotation center (IRC) is defined by intersecting the virtual wheel axis and the middle wheels’ axis. The position of the IRC and the relative distances to all the wheels is used to compute the required commands for all the motors. The corresponding set points, $\eta_i$ and $\omega_i$, for the six wheels are derived from $\eta_v$ and $\omega_v$ using the geometry of the rover.

Figure 7: CRAB velocity control based on the virtual wheel. The IRC position is defined as the intersection of the middle wheels axis and the virtual wheel axis.
3.4 Learning

Figure 8: Schematic representing the learning and prediction mechanism in the representation chosen for this work (using subspaces). The metric $\Lambda$ is the function that drives how the path is planned, and can be related to the rover’s behaviour as expressed in section 2.1.

This section describes the implementation of the required elements for the learning task. An illustration of this process is given in Fig. 8. The subspaces used in this work are the following. The first two represent the RTI types, the so-called softness and bumpiness which are local subspaces. The last subspace represents remote data and is called appearance, as linked with camera images. The creation of samples $s_i$ associating local and remote features is based on a 2D grid-based decomposition of the environment. The grid considered here is $G_l$, formed of cells $c_l$, which are used for the learning part.

For the remote features, the acquisition process is discrete and features are computed each time an image is taken. Thus the grid $G_l$ is projected onto the image obtained and the areas (called patches) defined by the visible cells $c_l$ are processed. The corresponding data, characterizing the remote aspect of the patches are saved within $c_l$.

The local features processing differs since it depends on continuous sources and their segmentation is driven by the wheels position. Thus, when a wheel moves into a cell the data of the local source is recorded until the wheel moves out, after which the local features are computed. Note that the wheels spend most of their time in different cells and each wheel is treated independently to generate the features. As an example, let us imagine the rover moving straight and parallel to one of the grid axes, the traversed cell ($c_l$) end up with three local features in each subspace. The three features are acquired sequentially as the three wheels enter and leave the cell.

When the the last wheel leaves the cell behind the rover, and the cell contains both local and remote features, a sample is created for use in the learning process. Note that ProBT$^5$, a programming tool facilitating the creation of Bayesian models and their reusability, is used to create and learn the needed probability distributions. We now continue with a description of the subspaces used, and their corresponding features.

$^5$http://www.probayes.com
3.4.1 Local feature: Softness

The local softness subspace, refers to whether the terrain has a soft soil (e.g. as sand) or a hard one (such as asphalt), which can be measured by the shocks occurring to the rover structure. Due to the particular suspension system of the CRAB rover, and its metallic wheels, every shock resulting from the wheel-terrain interaction is directly transmitted to the chassis. Thus the shocks can be measured on the chassis with the IMU. As this specific IMU measures the Euler angles, $\phi_j$ with $j$ referring to the three axes, one needs to process its data to retrieve the acceleration. As we are interested in the magnitude of the shocks, the absolute value of the angular acceleration along the $x$ and $y$ rover axis are used as features:

$$F_{l,softness} = (\Vert \alpha_x \Vert, \Vert \alpha_y \Vert)$$

where $\alpha_j$, $j = x, y$ is the angular acceleration of the robot. Note also that as $\Vert \alpha_j \Vert$ is a vector, the mean value is simply computed over the sequence corresponding to the grid cell treated. The angular acceleration along the $z$ axis is not used as it is coupled with changes in the heading of the rover.

This RTI design is inspired by other research work, such as (Brooks and Iagnemma, 2005) and (Weiss et al., 2006). These researchers have used this metric to classify the terrains the rover is traversing according to predefined classes. In these cases more complex features are used, such as the power spectral density (PSD) of the signal.

Finally, it must be noted that the data generated by the IMU is from a single source and expresses a RTI potentially influenced by all six wheels. In order to create the samples $s_i$, the IMU data must be related to the grid cells corresponding to the position of all six wheels.

3.4.2 Local feature: Bumpiness

The bumpiness subspace, which is also local, refers to the geometric characteristic of the terrain, whether it is flat (e.g. grass, at least on a soccer field) or bumpy (e.g. as sand). This characteristic can be measured by the movement of the CRAB’s suspension system, whose state is sensed by the angular sensors. Thus the $MR_b$ angles ($\tau$) are used to obtain the data of the suspension system movement. The average of the absolute value of the angular variation is proportional to the amount of displacement of the suspension system, which is defined as the bumpiness.

$$F_{l,bumpiness} = (\Vert \tau - \mu_x \Vert)$$

As each suspension mechanism has two redundant $MR_b$ sensors, a single data source can be computed as follow for the left - and similarly for the right - suspension system:

---

6The IMU Xsense MT9-B can provide either the Euler angles or the accelerations.
\[ MR_{b}^{left} = \frac{MR_{front}^{left} - MR_{back}^{left}}{2} \]  

Since in this case the sensors are quite noisy, a low-pass filter is also used to smooth the data:

\[ \tau_i = a \cdot MR_{b}^{left} + (1 - a)\tau_{i-1} \]

A value of \( a = 0.3 \) was empirically determined to solve the sensors’ noise issue. Note that the data of the \( MR_{b}^{left} \) and \( MR_{b}^{right} \) are indifferently mixed in the subspace as the structure is perfectly symmetrical. This leads to similar interpretations of the data if the metric is computed on the \( MR_{b}^{left} \) or \( MR_{b}^{right} \) signal.

In fact the bumpiness is fed by two continuous sources (\( MR_{b}^{left} \) and \( MR_{b}^{right} \)). This feature is specific to the CRAB rover. Due to its highly compliant suspension mechanism which adapts naturally to the terrain’s shape, the suspension movement directly reflects the roughness of the terrain.

### 3.4.3 Remote feature: Appearance

Although described last, the appearance subspace is not the least since it is the only remote one, and hence fundamental and necessary to predict the RTIs. The idea here is to be able to visually identify the different terrains using a colour based criteria. The Hue-Saturation-Value (HSV) colour space is usually preferred since it is robust to illumination changes. Thus (Brooks and Iagnemma, 2007) used, among other elements, a colour based terrain descriptor consisting of four values: S, V and the sine and cosine of H (to avoid any discontinuity problem). Such features were implemented and tested, but, in the context of our approach, they leads to an unreliable representation of the asphalt terrain (probably due to its gray colour which has a poorly defined Hue value.

In the present work, the appearance subspace makes use of a new colour representation, which is normalized and inspired from HSV, using the following features:

\[ F_r^{appearance} = (\Delta RG, \Delta GB) \]

with

\[ \Delta RG = \frac{R - G}{v} \]

\[ \Delta GB = \frac{G - B}{v} \]
\[ v = \max(R, G, B). \]

The divisor is named \( v \) as it is exactly how the \( V \) value is defined in the HSV colour space. This operation imposes a fixed \( V \) value and sensibly solves the problems with changing illumination. The difference between the values \( R \) and \( G \), and \( G \) and \( B \), are enough to describe the color then. This representation is different from the normalized RGB colour space due to this \( V \) value removal, which is a measure of where a particular color lies along the lightness–darkness axis. Removing this parameter allows the feature to be more robust to changes of illumination.

### 3.5 Prediction

This subsection concentrates on the use of the predicted \( M_{RTI} \) and the path planner \( E^* \). As both the learning part and path planning part are based on grid-based methods, the interaction is easy to foresee. In fact the cell sizes of both grids are subject to constraints that are different and contrary. On one side, the learning grid \((G_l)\) cells’ size must be sufficient to log a portion of signal form the local sources that is significant. On the other side, the path planning grid \((G_e)\) cells must be fine enough to drive the rover efficiently. For these reasons, the two grids are aligned but have different resolutions. The details of both are presented in the following parts.

#### 3.5.1 Grids Sizes

According to the position and orientation of the rover, the grid is overlaid onto every new image taken. Then the image is divided into patches corresponding exactly to the cells and the visual features are computed and stored into each \( G_l \) corresponding cell. The local sources are handled based on the position of the wheels as previously explained. The size of the grid is defined as the rover’s radius, which is slightly larger than half its width:

\[ S_{G_l} = \frac{\text{Length}_{\text{tot}}}{2} = \frac{0.196 + 0.648}{2} = 0.422 \text{ m} \]

\[ S_{G_e} = \frac{S_{G_l}}{\lambda_{le}} \]

However, such a grid size is not fine enough for the planning and therefore a resolution \( \lambda_{le} \) times higher is used for \( G_e \). In our implementation, \( \lambda_{le} = 5 \), which corresponds to an \( E^* \) resolution smaller than 10 cm. Such a resolution is sufficient for accurately planning the rover trajectory while having a reasonable computational cost.
3.5.2 Differential Prediction

The visual features \( F_r \) are computed for each one of the patches extracted from the images. The patches correspond to a portion of the image defined by the projection of the \( G_l \) cells on the image. Based on those \( F_r \) and according to equation 10, the most probable \( F_l \) are asserted for each one of the cells, and correspond to the expected RTI. An additional predicted cost \( r^p(c_e) \) can be computed for each one of the observed cells of \( G_l \), adjusting the E* propagation costs of the corresponding \( G_e \) cells, as follows:

\[
 r(c_e)^+ = r(c_e) + r^p(c_e) \tag{26}
\]

The idea is to adapt the cells’ cost \( r(c_e) \) by a small amount (\( \epsilon \)) each time a prediction can be performed. \( \epsilon \) is either positive when the predicted RTI metric (\( M^p_{RTI} \)) is ”good” or otherwise negative. As a reminder, note that the meaning of a ”good” or ”bad” RTI is assumed to be provided by the user. Hence, let us assume the RTI performance is measured by the softness and that softer soils are preferred. In this case, a ”good” RTI corresponds to preferring lower values of the feature expressing the softness. The additional predicted cost (eq. 27) is then computed as follow, based on the RTI experienced from the beginning of the test (eq. 28).

\[
 r^p(c_e) = \begin{cases} 
 +\epsilon & \text{if } M^p_{RTI} < M^m_{RTI} \\
 -\epsilon & \text{if } M^p_{RTI} > M^m_{RTI} \\
 0.0 & \text{if } M^p_{RTI} = M^m_{RTI} \text{ or if unknown} 
\end{cases} \tag{27}
\]

with:

\[
 M^m_{RTI} = \frac{M^{\max}_{RTI} + M^{\min}_{RTI}}{2} \tag{28}
\]

The additional predicted cost is positive when the RTI predicted (\( F_l^p \)) corresponds to a characteristic smaller than the median RTI encountered during the test (until then) and negative otherwise. As a practical detail, \( \epsilon = 0.05 \) is used during our experiments.

3.5.3 Prediction Processing

Two steps are performed prior to integrating the additional predicted cost \( r^p(c_e) \) within E*, as suggested by eq. 26. First of all, a step dilating the additional predicted cost is needed. During the learning process, the RTI is computed associating the images patches with the wheels’ position. On the other hand, E* plans a path as a trajectory to be followed by the center of the rover. Therefore, the cost expected by E* has to characterize the RTI of a rover center and \( r^p(c_e) \) must be updated accordingly. From the point of view of prediction,
Figure 9: Different steps of the terrain prediction. Original prediction (left), dilated prediction (middle) and the resulting E* grid (right). Note also that on the latter, the starting position and goal are depicted with the green dots.

The additional predicted cost corresponds to a rover whose position and orientation places one of its wheels into the cell. By considering the position of the wheels with respect to the rover’s center, this problem is overcome by expending the worst additional predicted cost, which corresponds to applying a dilation mask of radius $\frac{\text{Width}}{2}$ on $r^p(c_e)$.

The second step corresponds to a simple smoothing operation. In fact, discontinuities in $r^p(c_e)$ would result in unnatural paths planned by E*. Thus, having positive and negative additional predicted costs create edges at the terrains’ borders and results in traces planned which tend to follow these discontinuities. In the end, the resulting traces contain sharp turns and are very abrupt. The use of a very simple Gaussian filter of size $\lambda_{le}$, corresponding to one cell $G_l$, solves this issue.

Fig. 9 depicts $r^p(c_e)$ on the left, the result of the dilation operation (middle) and the resulting E* grid (right). The additional predicted costs sent are based upon the principle of trying to minimize the vibrations within the rover’s structure. Following this rule, grass is better than asphalt. The specific situation depicted here corresponds to Fig. 10. The grass, on the left hand side, leads to a positive $r^p(c_e)$. The asphalt terrain receives, on the contrary, a negative one.

### 3.6 Summary

The entire implementation of the current approach is described in this subsection and is summarized in pseudo-code corresponding to the whole software depicted on Fig. 2, and results can be observed on Fig. 10. The rover is placed in a situation where two terrains can be observed. These two terrains where previously encountered by the rover and therefore, the RTI can be predicted ahead of the rover. The green and red areas correspond, respectively, to positive and negative $r^p(c_e)$. The blue patches are unknown (or class 0). Note the dual resolution that is used to refine the border between different areas. When a cell $c_l$ is classified as unknown and surrounded by two different known neighbors, the image patch is divided into subpatches corresponding to its cells $c_e$. Those subpatches are processed to classify them and predict their RTI. This ensures a good detection of the transition of the two terrains type. Note that the learning part is not affected by this procedure and does not take the subpatches into account.
It can be noted that the prediction of the local features is pretty straightforward, especially since there is only one remote subspace; however it is not difficult to use several remote subspaces since their final prediction would be the result of naive Bayesian fusion.

Figure 10: Original image taken from the CRAB (left) and its terrain cell decomposition and prediction using two resolutions (right). The overlay corresponds to the additional predicted cost attributed.

**Input:** Pose<sub>rover</sub>; Data ← Pose estimate and Sensors data  
**Output:** Ctrl<sub>rover</sub> ← Rover commands  
**while** Waypoint not reached **do**  
  Update the rover’s pose within G<sub>t</sub> and G<sub>e</sub>;  
  Compute d, the distance to the last position where an image was taken;  
  **if** cell containing wheel pose changed from c<sub>t,i</sub> to c<sub>t,j</sub> **then**  
    Compute RTI, of c<sub>t,i</sub> → F<sub>t,i</sub>;  
  **end**  
  **if** d > S<sub>G</sub>, then  
    Take an image;  
    Extract patches from image (n<sub>patches</sub>);  
    Compute F<sub>r,n</sub> with 1 ≤ n ≤ n<sub>patches</sub>;  
    Predict RTI → F<sub>r,n</sub> = f(F<sub>r,n</sub>);  
    Compute r<sub>p</sub>(c<sub>e</sub>), predicted additional cost;  
    Process r<sub>p</sub>(c<sub>e</sub>), dilate and smooth;  
    Send r<sub>p</sub>(c<sub>e</sub>) to E* → r(c<sub>e</sub>) = r(c<sub>e</sub>) + r<sub>p</sub>(c<sub>e</sub>);  
    Update E* trace;  
  **end**  
  **if** c<sub>ln</sub> moved behind the rover **then**  
    Create s<sub>ln</sub> corresponding;  
    Save s<sub>ln</sub> for learning;  
  **end**  
  Compute Ctrl<sub>rover</sub> based on trace;  
  Send Ctrl<sub>rover</sub> to the rover control module;  
**end**  
**Algorithm 1:** RTILE pseudo code
4 Results

This section is dedicated to the description of the tests performed, their focus and the corresponding results. Four tests are presented here showing different aspects of the approach. The first and second tests illustrate the learning capability of the approach, in both controlled and natural environments. The third test shows the action of the whole method in a controlled environment, whereas the fourth test is performed in a more realistic environment.

4.1 Learning: Controlled Environment Trial

This test is designed to show the learning capabilities based on the rover’s knowledge. In this context, the three subspaces presented in the previous section are used. The RTILE approach enables the rover to learn its RTI models in an unsupervised manner within each subspace. Which means that the classes learned are not based on a human defined criterion. Therefore, it can be difficult to show the learning capabilities if the output is not understandable. For this reason, this first test places the rover in a controlled environment where its RTIs are sufficiently distinctive and expected to be learned into different classes. This experiment is also referred to as test 1.

4.1.1 Setup

The tests were performed indoors, using carpets to control the terrain appearance (Fig. 11). The Softness of the terrain can be controlled by the number of carpet layers used while the bumpiness is changed by using wooden stick placed under the carpets. Thus the appearance, the bumpiness and softness could be defined independently. The test consists of four successive runs labeled A to D. In each run, the rover is placed in a situation with a given Appearance, Bumpiness and Softness. The rover is then commanded to move forward on a five meters long trajectory and is expected to learn correctly from the data logged during the run. The rover moves at a speed of $10 \text{ cm/s}$. According to Fig. 2, the near to far and learning parts are used, whereas the path planning and the prediction part are dismissed. Except for the first run, where no prior information is available for the learning algorithm, the knowledge resulting from the learning performed on the previous runs is available.

The following runs are performed and the associated terrain can be seen in Fig. 12:

- In the run A, the rover moves on the concrete, hardest surface, which is flat.
- In B, a single layer of grass-like carpet is used. The terrain is flat.
- For run C, a brownish carpet with wooden sticks is used. The wooden sticks are $2.4 \text{ cm}$ in height and $4 \text{ cm}$ in length and are placed $70 \text{ cm}$ apart from each other.
- Finally in run D, the grass-like carpet is reused. The terrain is flat, but this test makes use of three layers of carpet.

In a first step, the sequence of runs A, B, C and D, named $ABCD$, is analysed in detail. Then the knowledge acquired after the four runs for a different sequence is also analysed. It is
interesting since the learning is based on processing the data on the basis of what is already learned. The different sequences analysed are the following: $ABCD$, $DCBA$, $CDAB$, $CADB$ and $BADC$. The first run of every sequence is performed without any prior knowledge. Note that the length of the trajectory is equal for each run, and the number of samples generated for each run is also equal. Thus, over a five meters trajectory, 18 samples are generated.

![Figure 12: Appearance of the various terrain types used for test 1. Run A (left) B, D (middle) and C (right)](image)

4.1.2 Results

The progress of the rover knowledge, resulting from the learning performed on the successive runs data of sequence $ABCD$ can be seen in Fig. 13.

The first run results in learning an initial class for each one of the subspaces, which is normal since no prior information is available at this point and therefore all the samples can only be labeled as unknown and used to learn the first class.

The second run presents a different appearance, as well as a softer terrain. As expected, a second class was learned in both subspaces but the bumpiness model remains as a single class, meaning that the bumpiness features were correctly classified.

Run C offers a different appearance, and a different bumpiness. Both were correctly learned and the softness is correctly recognized as equivalent to the previous test (both) have only a single layer of carpet.
Finally, the last run has only a softer surface, but it was correctly learned since a third softness class is added at this point. Both the appearance and the bumpiness are correctly interpreted as previously learned.

The final distributions have the data depicted in Fig. 14. The class number is written beside the corresponding distribution, followed by the mean values of the corresponding features. Note also that the whole features space is not shown.

<table>
<thead>
<tr>
<th>Run</th>
<th>Appearance</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Class 1</td>
</tr>
<tr>
<td></td>
<td>No prior</td>
</tr>
<tr>
<td>B</td>
<td>created</td>
</tr>
<tr>
<td>C</td>
<td>created</td>
</tr>
<tr>
<td>D</td>
<td>created</td>
</tr>
</tbody>
</table>

Figure 13: Progress of the rover knowledge during the first test sequence, $ABCD$. The softness (left) and bumpiness (right) subspaces can be seen in parallel, or independently. Their number of elements is alike as it depends on the appearance classes.

The rover knowledge, resulting from the learning performed for the different sequences can be seen in table 3. It shows that the different sequences end up with very similar results, but with little variation. The sequences $DCBA$, $CDAB$ and $CADB$ have a marginal additional class appearing in the softness subspace while the sequences $BADC$ has one appearing in the bumpiness subspace.
Table 3: Test 1 classification of samples.

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Appearance</th>
<th>Softness [%]</th>
<th>Bumpiness [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>ABCD</td>
<td>1</td>
<td>100.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.0</td>
<td>50.0</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.0</td>
<td>100.0</td>
</tr>
<tr>
<td>DCBA</td>
<td>1</td>
<td>100.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.0</td>
<td>44.4</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.0</td>
<td>100.0</td>
</tr>
<tr>
<td>CDAB</td>
<td>1</td>
<td>100.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.0</td>
<td>44.4</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.0</td>
<td>100.0</td>
</tr>
<tr>
<td>CADB</td>
<td>1</td>
<td>100.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.0</td>
<td>44.4</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.0</td>
<td>100.0</td>
</tr>
<tr>
<td>BADC</td>
<td>1</td>
<td>100.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.0</td>
<td>50.0</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.0</td>
<td>100.0</td>
</tr>
</tbody>
</table>

4.1.3 Summary

The learning approach is successfully validated in this test and several elements can be highlighted. First of all, the sequence ABCD, presented in more detail, shows interesting elements. A new class is learned each time it is expected. Then, on the other hand, the previously learned classes are correctly recognised if the rover is confronted to a characteristic that was learned. This can be seen for each one of the subspaces, the appearance (D), the bumpiness (B and D) and the softness (C). This proves that the features in each of the subspaces are good. The approach is also capable of relearning as the inference of the softness has evolved after the fourth run. The subspaces being learned independently, hence the approach is flexible.

Second, considering the different sequences tested, Table 3 shows that the different sequences end up with the same results, with very few variations, which proves that the features used characterize the terrains well and that the knowledge acquired is meaningful.

Also note that the classification of the different runs into the expected classes is the result of well defined subspaces and a specifically designed experimental setup. The subspaces are well defined because they recognize specific terrain characteristics and their impact on the rover. The different runs were designed with knowledge of the subspaces to achieve the terrain representation. It is obvious that using different subspaces (e.g. using other sensors) or using different setup would result in a different terrain representation.

Finally, as Fig. 14 shows, the learned distributions of the appearance subspace are peaked.
Figure 14: Representation of the learned distributions. **Smoothness** (upper left) **Bumpiness** (upper middle) and **Appearance** (bottom). Note that the whole feature space is not represented. In each plot, the classes’ numbers are written close to the corresponding probability distribution, in addition to the mean values.

This corroborates the approximation assumed in eq. 8. Therefore the learning method used in the context of this work offers all the required properties.
4.2 Learning: Natural Environment Trial

This test is presented to analyze the capabilities of the learning aspects of the approach in a real environment. Therefore, it is very similar to the previous test except for the lack of clear ground truth regarding what has to be learned and how. Using a natural environment allow the algorithm to really learn according to its own terrain representation. Nevertheless the results can be analyzed in terms of segmentation and sensitivity. This test is also referred to as test 2.

4.2.1 Setup

The CRAB rover is driven manually over a 310 m long trajectory at a speed of 10 cm/s in the test environment of a little village in the countryside. During the test, the rover was driven on various terrain types such as thick grass, gravel and asphalt. The transitions in between these types can also be counted as separate terrains. The rover trajectory and information regarding the terrains can be observed in Fig. 15. Note that the surface encountered by the rover is generally flat.

![Trajectory of the CRAB rover during test 2](image)

Figure 15: Trajectory of the CRAB rover during test 2. Different types of terrains as well as the transition in between can be observed. The terrain classes leave a lot of room for interpretation, since the ground truth is difficult to define.

4.2.2 Results

The test has the following result with respect to the learning algorithm. We’ll focus here on the appearance subspace, as it is then the entry point to differentiate terrains and reuse the knowledge acquired.

In the first section, the algorithm learned the grass as the first class. The second class learned is the transition between the grass and the gravel. Then the gravel resulted in learning a new class, class number three, but this class also includes asphalt, which is not distinguished from gravel. In the gravel section, some small patches of grass resulted in learning a fourth class. At the end of the test, all the patches acquired from the images are classified. These results can be observed in Fig. 16.
Regarding the RTI subspaces, the softness proved to be very sensitive as no less than 11 classes were obtained from the test. The bumpiness resulted in only two classes, but this can be expected since the CRAB did not faced any obstacle whatsoever. The detail of the probability distributions learned from this test can be observed in Fig. 17.

### 4.2.3 Summary

The learning algorithm works well and the results for each subspace are good. In the appearance subspace, the asphalt terrain is not distinguished from gravel due to the lack of features representing the texture. On the contrary, although the features used are fairly simple, a lot of different terrain types are ”seen” in the softness subspace. This experiment shows that it is important to use the right features to represent the terrain in such an unsupervised and generic approach.

As for the previous test, and as Fig. 17 shows, the learned distributions of the appearance subspace are peaked. This shows again that the approximation of eq. 8 is well founded.
4.3 RTILE: Controlled Environment Trial

The test is designed to verify the validity of the whole approach and especially the rover’s behaviour influence based on the knowledge available and learned during the test. This implies that we need to choose the function which rates the RTI, $M_{RTI}$, and also depends on the test environment. These tests are aimed at showing the impact of the overall method on the rover control, so the function is chosen to minimize the amount of vibration within the structure, thus defining:

$$M_{RTI} = \sqrt{F_{l,1}^{softness^2} + F_{l,2}^{softness^2}}. \quad (29)$$

Therefore, in this test, the path is planned according to the softness prediction of the terrain ahead of the rover. Note that the bumpiness is still acquired and handled, but it is not taken into account to evaluate the $M_{RTI}$ driving the rover’s path.

Following the successful results of the learning trial and in order to validate the approach step by step, the present test is performed indoors, in controlled environment. Every component of the RTILE approach described in section 2 is used. This test is also referred to as test 3.
4.3.1 Setup

The rover is initially commanded to move towards a waypoint four meters ahead, and then another ten meters ahead of the starting position. The configuration of the environment can be seen in Fig. 18, and two types of terrain can be observed. The first terrain type is concrete whereas the second is grass-like carpets. The straight trajectory between the starting position and first waypoint drives the CRAB mostly on the grass-like carpet and a straight trajectory between the first waypoint and the goal is only on concrete. Such straight trajectories can be considered as the default trajectories since the rover would move along them to reach the waypoint without taking the environment into account. The rover starts the test with a prior regarding the concrete and its corresponding softness, which means that the rover was driven for a few meters on this surface and it was able to characterize it. Therefore, concrete is "known" at the beginning of each test run. The test is repeated five times and the rover drives autonomously at a speed of $6 \text{ cm/s}$.

![Rover trajectories](image)

Figure 18: Setup of second test and resulting trajectories

4.3.2 Results

The first section of the test, which reaches the first waypoint, is performed in a straight line. The prior is not sufficient to challenge the obvious straight movement on a surface which is completely unknown. Reaching the waypoint, the samples acquired are processed and learned. This results in the creation of a new appearance and softness class corresponding to the grass-like carpet. In the second section of the test, the default trajectory is challenged by the opportunity of moving almost all way through on the grass-like carpet, which is softer than the concrete. As shown in Fig.18 the rover takes a slightly longer path to remain on the carpet.

In order to express what happened during the test, it is necessary to defined metrics summarizing the rover behaviour. Thus, two metrics are defined here, one corresponding to the distance traveled and the other corresponding to the softness of the terrain.
\[ M^{Dist} = \sum \sqrt{\Delta x^2 + \Delta y^2} \]  \hspace{1cm} (30)

\[ M^{Soft} = \sqrt{\|\alpha_x\|^2 + \|\alpha_y\|^2} \]  \hspace{1cm} (31)

with \( i \) being the trajectory section, between waypoints \( i \) and \( i + 1 \). As this test is composed of two sections, the first one being the same for the RTILE approach as well as the default approach, three pairs of results are presented in table 4.

<table>
<thead>
<tr>
<th>Section</th>
<th>( M^{Soft} ) [rad/s^2]</th>
<th>( M^{Dist} ) [m]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Default</td>
<td>Mean</td>
<td>0.14</td>
</tr>
<tr>
<td>RTILE</td>
<td>Std Dev</td>
<td>0.02</td>
</tr>
</tbody>
</table>

**4.3.3 Summary**

The softness of the second section has a value significantly lower using the RTILE approach. This is a success as a terrain minimizing the chassis vibrations is preferred. This result is consistent and repeatable, as the very low standard deviation over \( M^{Soft} \) shows. Fig. 18 indicates that the trajectories of the five runs have globally the same shape, but with some variation, which is reflected by the slightly higher standard deviation over \( M^{Dist} \), corresponding to the rover trajectory in the end of section 2.

When using RTILE, section 1 and section 2 correspond to the same terrain (grass-like carpet) but their \( M^{Soft} \) value is quite different. This difference is induced by the very last part of the trajectory, just before the goal, where the rover has to move a couple of meters on concrete. This raises the value of the \( M^{Soft}_{2,RTILE} \).

In despite of these variations, the paths show that the rover behaviour is influenced by the learned RTI and this knowledge allows the path of the rover to be adapted. The E* path planner is successfully used and integrated to the approach and the inference of the RTI on the terrain is successful.

Finally, to have an insight into the test, the map rating the RTI in this environment, as well as the processing of an image to recognized previously met terrain are shown in Fig 19.
Figure 19: Final prediction map (left) and prediction on the carpet (right). The prediction map shows the value of $\Lambda$ for each one of the observed $c_i$, which corresponds to the sum of the multiple rewards computed. The area depicted corresponds only to the second test section. The first bringing nothing as all the $F_i$ are unknown at this time. The second carpet can be well identified on the prediction map (brownish area). On the right hand-side, the prediction performed during the second test section, on the carpet is shown. The result is very satisfying as only the transitions cells are unknown.

4.4 RTILE: Natural Environment Trial

The test is very similar to test 3. It aims also at verifying the entire approach, but a step further by using a natural environment. Here, natural means that the environment is not as controlled as before, and some additional uncertainties are therefore taken into account. Therefore, the same $M_{RTI}$ function is used as with the previously presented trial. Thus, the path is planned according to the softness prediction of the terrain ahead of the rover, focusing on minimizing the vibration within the rover chassis. Note that for the same reasons as for the previous test, the bumpiness is not taken into account. This test is also referred to as test 4.

4.4.1 Setup

The test is divided in two parts. The setups of both parts are depicted on Fig. 20.

In the first part, the rover is commanded to reach a first waypoint five meters ahead, and then another 10m ahead of the starting position. Performing a straight, direct trajectory, the rover moves first on asphalt, and then on grass. In the second part, the rover aims for a goal positioned eight meters ahead and a straight trajectory would drive the rover on asphalt only. The test is also repeated five times to have more reliable results and the rover drives autonomously at a speed of $6 \text{ cm/s}$.

4.4.2 Results

The first part is executed without a prior and the waypoints are reached via a straight trajectory. It gives the opportunity to learn the asphalt and grass terrains, both in terms of
appearance and softness. The second part of the test makes use of the knowledge acquired in the first part. In this case, and as depicted in Fig. 20, the default trajectory is not used and a trajectory moving on the grass terrain is preferred. The resulting metrics of this test are presented in table 5. Fig. 21 shows the E* propagation costs, $r(c_e)$, resulting from the test.

<table>
<thead>
<tr>
<th>Table 5: RTILE natural environment (test 4) results</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\theta_{Soft}$ $[rad]$ $\theta_{Dist}$ $[m]$</td>
</tr>
<tr>
<td>Default</td>
</tr>
<tr>
<td>Mean</td>
</tr>
<tr>
<td>Std Dev</td>
</tr>
</tbody>
</table>

4.4.3 Summary

As the results above show, the use of RTILE allows the rover to select a softer terrain to reach the goal. As a side-effect, this increases the distance traveled since the trajectory is longer.

This test shows again the rover’s behaviour influenced as a result of the learned RTI, which influences the path of the rover. The test started without any prior knowledge regarding the rover behaviour; a short five meter trajectory on both terrains allowed the rover to sufficiently learn both terrains.
Figure 21: Final prediction map of test 4 showing the value of $\Lambda$ for each one of the observed $c_l$. The grass terrain is well recognized and predicted. It corresponds to the brownish zone.

4.5 Lessons Learned

The learning trials show that it is possible to learn from the rover’s experiments, which can be done without any prior and in an unsupervised way, except for the definition of the cost regarding the learning part. First, the features have to be designed with care, to represent a specific and meaningful RTI model. As the results of test 2 show, the features must also be sensitive enough to be able to express different RTI models according to different terrain types. Second, in order to have reasonable computational time and memory usage, the features spaces must necessarily have a reduced number of dimensions, which also underlines the need for carefully designed features. To summarize, using an unsupervised approach requires the transfer of much more knowledge within the features. From a general point of view, it can be also noted that too many classes are generally better than too few classes in the subspaces. A finer representation of the terrains encountered is achieved by having more classes modeling the RTI.

The two RTILE trials, show that the knowledge acquired to qualify the RTI can be used to influence the rover behaviour, or its path. Using RTILE has a cost regarding the rover behaviour, as the resulting trajectory is longer in both tests. This raises the following question: To what extent can the rover trajectory can be influenced by the RTI? And what is the maximal deviation from the default path in order to have an optimal rover behaviour?

Another point is that the remote subspaces are particularly important and are a key element of RTILE. They are the entry point to use the knowledge acquired and make any prediction as they are used to recognize the previously met terrains. If the appearance subspace, in the context of this article, regroups too many terrains type within the same class, the RTI predicted does not give any useful information. Then, the remote features classes are better being too specific, rather than too largely defined.

Finally and similarly to the previous point, the camera’s field of view has a big influence over
the RTI predicted. The camera used in this research has a reduced field of view and therefore, all the information available in the images is used to predict the RTI. In this context it is very important to be able to detect as precisely as possible the transition between different terrains, and in this perspective, the use of a dual resolution grid improves the results.
5 Conclusion

The Rover-Terrain Interaction Learned from Experiments (RTILE) approach aims to learn the Rover-Terrain Interaction (RTI) based on the rover’s experiments, without any prior. The knowledge acquired is subsequently used to form RTI predictions on terrains lying ahead of the rover, which results in an influence on the rover trajectory. The approach is implemented and successfully tested on the CRAB rover.

The near to far technique generates samples associating remote and local data, whose probability distribution can be learned with the ProBT tool. The E* path planner, combined with the predicted RTI, allows the rover’s behaviour to be influenced. The notion of subspaces, representing various RTI models is also introduced and three of them, namely Softness, Bumpiness and Appearance, are presented.

Finally the tests conducted with the CRAB rover show the feasibility of the approach as well as its successful implementation. The unsupervised learning indicates good results in detecting new classes within subspaces and the entire approach is successfully tested both in controlled and natural environments. The tests presented in this article show the interest and versatility of the approach.

The authors would like also to point out that the proposed method in this article is not foreseen as a replacement to reduce design activity or to get rid of any wheel-ground interaction model or even to avoid considering the terrain traversability. The approach proposed here has to be considered as a complementary tool, a performance enhancer for a well designed rover.

A few words about slip are necessary, as using it as a subspace can be very appropriate. Slip is important as several aspects of a rover mission demand for as little slip as possible. Firstly, navigation is more accurate if the rover does not slip. Secondly, since slipping wheels do not contribute to the rover’s movement, slip is a loss of energy. Finally, potential slip increases the risk of an operation failure due to loss of control of the vehicle. Despite all these elements, slip is not considered in this work and this is mainly due to two reasons.

- The first reason is that slip can only be measured if a reference (or ground truth) is available. This is not the case for the CRAB rover (but is being developed).
- The second is that the goal of the current research is to show the rover under the influence of RTI models learned from its experiments, and to show a methodology to do this. In this context, the type of RTI models used are not as important as the approach itself. Thus, the authors do not feel compelled to use slip in this approach.

Hence forth, although the approach description is sound and the tests corroborate its potential, some questions are still open. Among them, two are of importance and are the focus of the upcoming work. The first one concerns the learning aspect. In the work presented here, all the new samples which could not be classified (or that are classified “unknown”) are used to generate a new class. The problem is that nothing proves that only a single new terrain was discovered and this is what is implicitly assumed. To solve this potential
problem, a clustering step has to be added before the learning. It can also be noted that this is not a critical problem if the learning process is performed more frequently than there are terrain changes in the environment. The second issue is linked with the characterization of the approach. It is known that E* path planner proposes a trade off between the path length and the trafficability cost (in the case of RTILE). In this context the values assigned to \( r(c_e) \) have to be characterized, answering the following questions: In an environment with several terrains, what is the impact of a change of \( r(c_e) \) on the E* proposed path? What is the relation between the variation of propagation cost and the amplitude of the difference between the RTILE and the default path? At the moment, it is assumed that the user provides RTILE with the metric \( M_{RTI} \) driving the cost. The next step is to refine the interaction with E* in order to be able to use an input such as: A detour of amplitude \( \delta m \) can be performed for a predicted \( \delta M_{RTI} \) between two terrains.
Acknowledgments

The authors gratefully acknowledge Mark Höpflinger and Majid Nili for their help and expert advices in the early phase of the project. Thank also to Dr. Francis Colas and Dr. Mike Bosse for their inputs regarding the redaction of this article. This project was partially funded by the European Space Agency (ESA). Finally, a special thank to our trustworthy Cunégonde and Bröiseli, who are always ready to give a hand.
References


