Estimating wheel slip of a planetary exploration rover via unsupervised machine learning

Justin Kruger
Stanford University
Stanford, CA, 94305
971-273-8878
jjkruger@stanford.edu

Arno Rogg
NASA Ames Research Center
Moffett Field, CA, 94035
650-604-1045
arno.rogg@nasa.gov

Ramon Gonzalez
robonity: worldwide tech startup
Calle Extremadura, no. 5, 04740
Roquetas de Mar, Almeria, Spain
ramon@robonity.com

Abstract*—Planetary exploration rovers often encounter imperfect traction and wheel slip, which negatively impacts navigation and in the worst case can result in permanent immobilization. Recent studies have applied machine learning to estimate rover wheel slip, which this paper extends via the implementation of three unsupervised learning algorithms: self-organizing maps, k-means clustering, and autoencoding. Unsupervised learning is preferred since labelled training data may be risky or time-consuming to obtain on site; each algorithm classifies the rover's current slip state into one of several discrete categories. Proprioceptive sensors are used to avoid added complexity and prevent a reliance on visual odometry. The algorithms are validated using sensor data from a planetary rover driving on a sandy incline, and performance is evaluated for different velocities, sensor inputs, slip classes, algorithm parameters, and data filters. Self-organizing maps (SOM) demonstrate the best slip classification accuracy, achieving 97% immobilization detection in the ideal two-class case. At rover-like speeds of 0.10 m/s, 88% accuracy is demonstrated for three classes. For ten slip classes, 71% accuracy is obtainable. Compared to SOM, k-means loses 5-30% accuracy and autoencoders lose 2-10% accuracy. SOM is most computationally intensive while k-means is least. An analysis of significant parameters for algorithm tuning displays accuracy benefits of up to 25%, and mis-classifications can be further reduced by modifying class boundaries. The algorithms are generic and can be trained for different terrain, environment or vehicle parameters, and although some labelled data is needed to directly associate unsupervised clusters with slip classes, it is significantly less than what a fully-supervised algorithm requires. Unsupervised learning is thus considered promising for robust real-time rover slip estimation.

TABLE OF CONTENTS

1. INTRODUCTION

Planetary environments such as Mars and the Moon are challenging terrain for rovers. Terrain is often very loose or steep, resulting in sub-optimal tractive performance and wheel slip. This negatively impacts rover localization and in the worst case may lead to rover immobilization. The Mars Exploration Rover 'Spirit' became permanently embedded in 2009 after its wheels could not gain traction in a sand dune, eventually ending its mission, while in 2005, the 'Opportunity' rover spent many weeks extracting itself from a similar event [1] [2]. NASA's 'Curiosity' rover opts to avoid high-slip areas by driving over rock, at the cost of increased wheel damage [3] [4]. Even though avoidance may be possible – or preferable – it is often necessary for rovers to navigate high-slip terrain in order to reach their scientific targets. Embedding avoidance then becomes a crucial aspect of rover safety, for which estimation of wheel slip is a key element [5] [6] [7] [8].

Wheel slip is generally defined as $s = (\omega r - v)/\omega r$, where ω is angular velocity of the wheel, r is its radius, and v is the linear velocity of the wheel's center [5]. Excessive slip can lead to vehicle slowdown, navigational inaccuracy, inability to reach objectives, or permanent entrapment. Thus, improving the ability of a rover to quantify its slip is vital. If rovers can independently detect slip when it occurs they will be able to travel with greater speed, safety and reliability; a primary characteristic of rovers is their autonomy, and more reliable slip estimation will allow future missions to achieve more ambitious scientific objectives without the current requirement of extensive human supervision [9].

Many different approaches to slip estimation have been suggested, employing various combinations of sensors and algorithms. In the simplest case, wheel velocities can be compared to an integrated accelerometer reading, but this is subject to drift [10]. Others have extended this by using EKFs to fuse encoder, IMU, and GPS information with a vehicle dynamics model [11]. While this can be accurate (even without GPS, which a planetary exploration rover cannot rely upon), performance is dependent on the accuracy of the model and knowledge of terrain parameters. Motor current measurements are often employed: Curiosity uses a high current threshold to detect embedding and past experiments have combined encoder, IMU and current measurements to detect all-wheel slip [12] [13]. However, this approach is less effective on slopes where current is naturally higher. Many of the most developed techniques for slip estimation rely on visual odometry (VO), tracking landscape features or a rover's wheel traces [7] [14] [15] [16]. This has proven effective, although performance is heavily influenced by terrain features and lighting. VO can also be computationally intensive and places a low limit on rover speed. Other exteroceptive solutions, such as localization via satellite or radio beacons, are impractical for a rover. Gonzalez and Iagnemma present a discussion of these and other approaches in their state-of-the-art survey [8].

In light of these difficulties, a more recent area of interest is machine learning (ML), with the goal of applying ML algorithms to discover patterns in proprioceptive rover sensor inputs and thus classify its slip state into discrete classes.

Proprioceptive sensors are preferred over VO as they are not dependent on lighting and terrain features and do not require computationally-expensive image processing. Gonzalez et al. obtained promising results with this approach, using Support Vector Machines, Self-Organizing Maps and Artificial Neural Networks to classify rover slip as low, medium or high [17]. Similar studies have implemented ML regression to provide a continuous slip estimate, or have applied data aggregation and Bayesian tracking in an unsupervised slip classifier that improves online [18] [19]. These initial explorations have found that ML techniques are fast, applicable to a wide range of environments, and can provide good accuracy.

This study builds upon this work by investigating three ML algorithms in more detail, evaluating their performance for rover slip classification. Specifically, it focuses on the less-investigated area of unsupervised learning. Although supervised learning will generally achieve better accuracy, such methods can be impractical for rovers as they require labelled training data, which could be risky or time-consuming to obtain on site [17]. The three algorithms under investigation are self-organizing maps (SOM), based on the promising results of Gonzalez et al.; k-means clustering (KM), as it is one of the most common unsupervised classifiers; and autoencoding (AE), which is an oft-used dimensionality reduction technique not yet applied to this context. By examining the performance of these algorithms for a greater range of parameters, a more definitive recommendation can be made towards the potential suitability of ML-based rover slip estimation. A discrete (as opposed to continuous) slip estimate is used as it is a much easier problem for unsupervised learning to deal with while still providing useful information to the rover [20].

2. UNSUPERVISED MACHINE LEARNING

Unsupervised algorithms aim to detect and exploit similarities in data to cluster similar inputs together – in this case, operating on a vector of rover sensor inputs to classify its current slip state. The algorithms in this study were implemented in MATLAB R2017a, running on a PC with an i7-5500U 2.4 GHz CPU and 8 GB RAM.

Self-Organizing Maps

The SOM algorithm manipulates a network of neurons. An initially-unordered neural network is gradually moulded into a topologically-ordered map of neuron clusters, emulating the way sensory inputs are mapped in the brain. A brief description of SOM is provided below, with additional mathematical detail available from Kohonen [21] and Obermayer and Sejnowski [22].

Let $\mathbf{q} = [q_1, q_2, ..., q_i]^\top$ be an input vector of i-dimensional data, and let $\mathbf{n} = [n_1, n_2, ..., n_k]$ be a list of k total neurons, arranged in a 2D grid. Denote the weight vector of neuron n_j by $\mathbf{w}_j = [w_{j1}, w_{j2}, ..., w_{ji}]^\top$, describing its position in the input space. First, neuron weights are randomly initialized. Then, $\mathbf{q}$ is compared to each $\mathbf{w}_j$ to find the best-matching neuron. Often, the L2-norm $c(\mathbf{q})$ between $\mathbf{q}$ and $\mathbf{w}_j$ is used. The 'winning' neuron is excited, along with its spatial neighbors, via a time-varying Gaussian function $h_{j,c(\mathbf{q})}$. Neurons within the influence of the winning neuron have their weights updated via:

$$\mathbf{w}_j(t+1) = \mathbf{w}_j(t) + \eta(t) h_{j,c(\mathbf{q})}(t)(\mathbf{q} - \mathbf{w}_j(t)) \quad (1)$$

where t is the iteration number, and η is a learning rate that decreases with time to encourage convergence of the map. It is also advantageous to begin with a large neighborhood that shrinks with time. In this fashion, input vectors are matched to neurons, which excites those neurons' neighbors and updates their weights. Over many iterations, neurons activated by similar inputs become topologically clustered. Clusters are separated by larger inter-neuron distances.

Network performance is evaluated by creating a network semantic map $\mathbf{M}$. First, a class must be assigned to each sample input used for training. Then, the class C of the sample $\mathbf{q}_m$ that best matches the weight of neuron n_j is assigned to node j in $\mathbf{M}$ [17]. For robustness, the final class of each neuron in the network is the mode of classes of the set of p best-matching inputs. The trained semantic map (i.e. neurons paired with labels) can then be used to return to class of any test input. This is simply the class of the neuron which the input stimulates (i.e. the neuron whose weights are closest to the input vector). Note that while SOM is unsupervised, some training data is required to create the semantic map.

K-Means Clustering

The well-known k-means algorithm partitions a set of data into k clusters [17]. Initially, k cluster centers are randomly positioned in the input space. A new point $\mathbf{x}_i$ is added to whichever cluster is closest. The mean of that cluster is then recalculated to take the new point into account. Thus, at every stage, the 'k-means' are the means of the clusters they represent. The algorithm iterates until cluster centers stop moving and points are no longer being reassigned to different clusters. There is no guarantee that the optimum will be found with this algorithm, so k-means is often run multiple times with the final result being the one that minimizes the distance between all points and cluster centers.

After creating clusters, performance can be evaluated by associating each cluster with a class (in this case, the mean class of the points that comprise it). Then, the class of an input vector is the class of the cluster whose center is closest. As with the SOM algorithm, KM is unsupervised, though some labelled training data is needed to directly associate each cluster with a slip class.

Autoencoders

Autoencoders are artificial neural networks that aim to compress data into a shorter, encoded form, which can later be uncompressed into something closely matching the original [23]. The simplest arrangement of autoencoder consists of an input layer of neurons, an output layer, and one or more connecting hidden layers. The output has the same number of nodes as the input so as to reconstruct the original data. The autoencoder – by learning a compressed representation – is discovering notable data features, which can be leveraged to separate inputs into classes. Autoencoders learn in an unsupervised fashion, in contrast to similarly-configured supervised networks such as multilayer perceptrons.

The encoding and decoding layers can be defined as transitions $\phi : \mathcal{X} \to \mathcal{F}$ and $\psi : \mathcal{F} \to \mathcal{X}$. For a single hidden layer, the encoder takes an input $\mathbf{x} \in \mathbb{R}^m = \mathcal{X}$ and maps it to $\mathbf{y} \in \mathbb{R}^n = \mathcal{F}$:

$$\mathbf{y} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b}) \quad (2)$$

Here, σ is an element-wise activation function (usually sigmoid or linear); $\mathbf{W}$ is a weight matrix; and $\mathbf{b}$ is a bias vector. Then, the decoder maps the image $\mathbf{y}$ to the reconstruction $\mathbf{x}'$ of the same shape as $\mathbf{x}$:

$$\mathbf{x}' = \sigma'(\mathbf{W}'\mathbf{y} + \mathbf{b}') \quad (3)$$

where $\sigma', \mathbf{W}'$ and $\mathbf{b}'$ for the decoder may differ from the encoder. Autoencoders are trained to minimize reconstruction errors, which is often the squared error averaged over some input training data set. If the feature space $\mathcal{F}$ is of lower dimension than the input space $\mathcal{X}$, the feature vector $\phi(\mathbf{x})$ can be regarded as a compressed representation of $\mathbf{x}$. Conversely, 'sparse' autoencoders have more hidden neurons than input neurons – to avoid over-activation of hidden units, only a small number are permitted to be active at one time, thus forcing hidden units to associate with specific data features [24]. Autoencoders can be layered to form different network structures and this paper applies two stacked sparse autoencoders: the first is trained on the input data, while the second is trained on the encoded features from the first. Additional layers did not improve results, likely due to the low dimensionality of the input data.

A final softmax layer is used to enable classification of inputs. This is a supervised training step in which a softmax function learns to 'squash' the k-dimensional vector $\mathbf{y}$ of autoencoder outputs into a k-dimensional vector $\sigma(\mathbf{y})$, for which entries are in the range $(0, 1]$ and sum to 1 [25]. The output is a probability distribution over k possible outcomes (i.e. k possible slip classes); the class of an input is the outcome with the highest probability.

3. Data Collection

Training and testing data was obtained from field trials of the ProtoInnovations Lunar All-Terrain Utility Vehicle (LATUV) [26]. The rover was driven up the sandy incline shown in Figure 1, along a 15 m course at fixed speed. Ten experiments were performed at velocities from 0.05 m/s to 0.25 m/s with two trials at each velocity. The rover was fitted with an RTK-GPS and an IMU near each wheel (Figure 2). Onboard telemetry included the positions and current draws of all drive motors and steering motors. Data was synchronized and logged at 10 Hz using custom ROS software [26].

Figure 1. Rover test environment.

To assess the accuracy of each ML algorithm, ground truth slip was determined by comparing the rover's RTK-GPS velocity to wheel encoder angular velocity. To combat noise, a median filter of length 10 (1 s) was applied to ground truth data. Negative slip values were ignored (and are the result of sensor noise). The following boundaries were assigned for different numbers of slip classes:

- 2 classes: [0, 0.3, 1]
- 3 classes: [0, 0.2, 0.5, 1]
- 4 classes: [0, 0.15, 0.35, 0.6, 1]
- 5 classes: [0, 0.1, 0.25, 0.45, 0.7, 1]
- 10: [0, 0.05, 0.1, 0.15, 0.2, 0.3, 0.4, 0.55, 0.7, 0.85, 1]

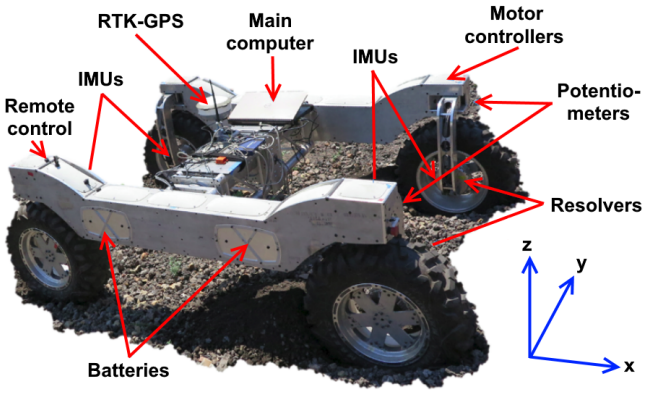


Figure 2. Rover hardware.

These classes and boundaries can be considered trade-offs between the accuracy of slip classification and the performance of an associated low-level traction controller, responsible for adjusting rover wheel velocities to achieve slip compensation. Two classes allows simple embedding detection, while ten classes approaches a continuous slip estimate. Further discussion and preliminary results from these controllers are presented by Gonzalez and Iagnemma in [20].

Two-fold cross validation was applied to randomly split the data into a training set (used to develop the behavior of each classification algorithm) and a testing set (used to test the performance of each algorithm). This ensures that algorithms are tested on unseen data. Since slip data possesses time dependence, at most three consecutive data points were assigned to the training set before the next data point had to be assigned to the testing set (or vice versa). Each result is the average of ten runs of an algorithm.

Sensor Inputs

Many sensor inputs are available for use, including 24 IMU measurements (6 DOF, 4 wheels) and 12 wheel readings (velocity, acceleration and current of each wheel). Different filters may also be applied, such as a sliding variance, median or mean. To investigate which inputs may be most appropriate or useful, the mean of the Pearson correlation, distance correlation, and maximal information coefficient [27] between each filtered input and ground truth slip was calculated. This calculation was performed across a union of all field trial data to avoid potential biases towards specific velocities. The highest combined correlation measures for various filters are presented in Table 1.

For maximum correlation, the set of motor current, x-acceleration (mean filters), z-acceleration and angular velocity (median filters) appears optimal, averaged over all wheels. The aforementioned work by Gonzalez et al. [17] also proposes the set of x-acceleration, z-acceleration, pitch rate (variance filters) and motor current (median filter) taken from a single wheel. Note that although wheel angular velocities are expected to be independent of slip, some correspondence was present, likely due to non-ideal motor behaviors. It is also useful to maintain independence between inputs to provide maximum information to the classifier and for this reason angular rates should be considered, despite being less correlated. A variety of input vectors are tested in Section 4.

To illustrate slip behavior, Figure 3 plots two normalized sensor inputs with rover ground truth slip for one test drive

Table 1. The average of the Pearson correlation, distance correlation and maximal information coefficient of different sensor inputs and ground truth slip.

#	Feature	Wheel	Filter	Correl.
1	Motor current	All	None	0.643
2	Wheel ang. vel.	Right rear	None	0.614
3	IMU z-acc.	All	None	0.603
4	IMU x-acc.	All	None	0.437
5	IMU y-acc.	Right front	None	0.411
6	IMU pitch rate	All	None	0.248
7	Motor current	Right rear	Variance	0.430
8	Wheel ang. vel.	Left front	Variance	0.326
9	IMU yaw rate	All	Variance	0.302
10	IMU z-acc.	All	Variance	0.284
11	IMU x-acc.	All	Variance	0.282
12	IMU pitch rate	All	Variance	0.265
13	Motor current	All	Mean	0.692
14	IMU z-acc.	All	Mean	0.652
15	Wheel ang. vel.	Right rear	Mean	0.643
16	IMU x-acc.	All	Mean	0.635
17	IMU y-acc.	Right front	Mean	0.532
18	IMU pitch rate	Right rear	Mean	0.488
19	Motor current	All	Median	0.689
20	IMU z-acc.	All	Median	0.657
21	Wheel ang. vel.	Right rear	Median	0.656
22	IMU x-acc.	All	Median	0.625
23	IMU y-acc	Right front	Median	0.538
24	IMU pitch rate	Left rear	Median	0.500

at 0.25 m/s using telemetry from the front left wheel. The median drive current is consistently correlated with slip levels, especially for medium and high slip – current rapidly increases as the rover attempts to advance and find traction. The variance of the yaw rate is roughly correlated with low and medium slip levels, whereas at high slip, the variance rapidly tends to zero. This is because the rover quickly embeds itself, resulting in much less motion. In this sense, different sensors and filters provide different information that may be leveraged by ML algorithms.

Algorithm Parameters

Each ML algorithm allows a variety of input parameters which can be tuned for optimal classification performance. During implementation, these parameters were investigated individually to observe their effects on classification accuracy for different rover velocities and slip classes. A set of near-optimal parameters could then be experimentally determined for each algorithm and data set, which were used to obtain the final training and testing results. This is discussed further in Section 4.

4. Results and Discussion

Sensor Input Selection

Table 2 presents classification accuracy for eight sensor sets. To avoid velocity effects, tests were performed on a union all field trial data using default algorithm parameters. Using mean sensor readings from all wheels provides a boost in accuracy, most significantly for k-means. For SOM, the

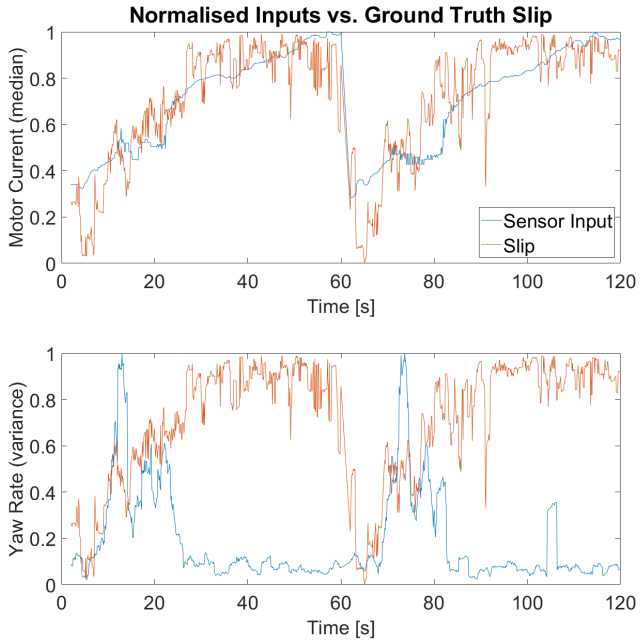


Figure 3. Filtered sensor measurements vs ground truth slip at 0.25 m/s. Both are normalized to lie between [0, 1].

choice of the most correlated inputs (as in the fourth set) provides a small boost to 66%. The best set of features was the fifth, giving 68% accuracy. This combination is not necessarily apparent from looking at correlation, which suggests correlation is not the only factor when applying SOM. Increasing the number of features (as in the fifth or seventh sets) has little impact and can even decrease accuracy, implying no useful information is being added. For k-means, some sets reduce accuracy by nearly 20%; the use of wheel angular velocity and yaw rate appears to confuse the KM algorithm. For AE, different sets have a negligible impact on performance. This implies the algorithm is robust but could be limited in its ability to leverage additional information besides motor current. To obtain final results, the best discovered input set for each algorithm was used. (Note, however, that more optimal sets may exist that were not tested here.)

Table 2. Classification accuracy for different inputs.

Sensor Inputs	Wheel	SOM	KM	AE
10, 11, 12, 19	Left rear	0.608	0.479	0.642
10, 11, 12, 19	Left front	0.617	0.481	0.646
10, 11, 12, 19	All	0.640	0.630	0.660
13, 16, 20, 21	All	0.658	0.630	0.654
9, 10, 11, 13, 18, 21	All	0.670	0.491	0.675
9, 10, 13, 21	All	0.678	0.449	0.667
8, 9, 13, 16, 20, 21	All	0.659	0.461	0.671
13, 20	All	0.598	0.630	0.655

Classification Accuracy

The accuracy of each algorithm for near-optimal parameters is displayed in Figure 4. The SOM method consistently gives the best classification accuracy, regardless of velocity or the number of slip classes. The KM method is consistently worst, while the AE method lies between them. For the easiest classification problem of 0.25 m/s with two classes, SOM

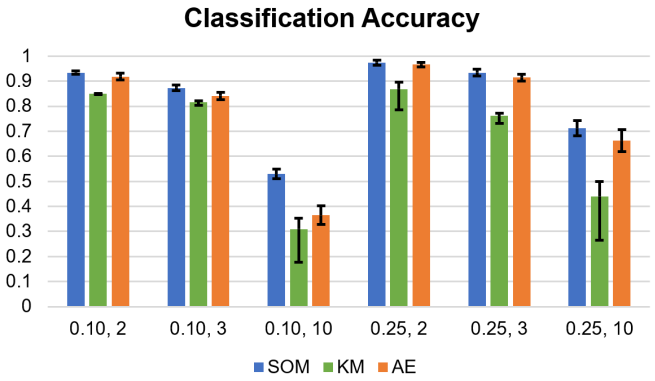


Figure 4. Average algorithm accuracy for different [velocity, # of classes] sets. Errors bars show the maximum and minimum accuracy over ten runs of the algorithm.

gives 97% accuracy. For the hardest problem of 0.10 m/s with ten classes, SOM gives 53% accuracy. The differences in accuracy between each method are more apparent with lower velocities and/or more classes.

Classification accuracy is almost always better at higher velocities. Much of the input data is drawn from IMUs, which measure vibrations in the chassis. These vibrations are largest at high velocities, whereas at low velocities, they are obscured by noise. Larger numbers of classes are also more challenging; data points are more likely to lie near a class boundary, making it difficult for the algorithm to assign a class with certainty.

Even at low, rover-like velocities (e.g. 0.10 m/s), the combination of unsupervised machine learning with proprioceptive sensing appears adequate for slip estimation. For pure immobilization detection (i.e. two slip classes), the >90% accuracy of SOM is comparable to or better than many previously-demonstrated techniques. In the ten-class case, which approaches a continuous slip estimate, SOM is able to classify slip with 53-71% accuracy. 'Adjacent' accuracy - i.e. classification into the correct class or an adjacent class - rises to 93%. This provides a great deal of information about the current slip state of the vehicle which could, for instance, be employed as part of a traction control system.

Due to random initialization procedures and cross-validation partitioning, outcomes are not fixed from trial to trial. The SOM algorithm is the most consistent, followed by the autoencoder, followed by k-means. In certain cases k-means possesses extremely high variability, due to the random way in which clusters are initialized and how they move as subsequent inputs are processed. For this reason, SOM is preferred.

Figure 5 displays ground truth slip with each algorithm's estimated slip class. In the first case (0.10 m/s), the KM and AE algorithms produce a less variable slip estimate with very few instances of medium slip. The SOM algorithm detects more instances of medium slip, resulting in slightly more accuracy. In the second case (0.25 m/s), SOM is again more likely to pick up small slip variations, more closely following the rover's actual slip behavior.

Figure 6 presents confusion matrices for the case of 0.10 m/s and three slip classes, using class boundaries of [0, 0.2, 0.5, 1] (left) and [0, 0.3, 0.6, 1] (right). For the default [0, 0.2, 0.5, 1] case, several strengths and weaknesses become evident

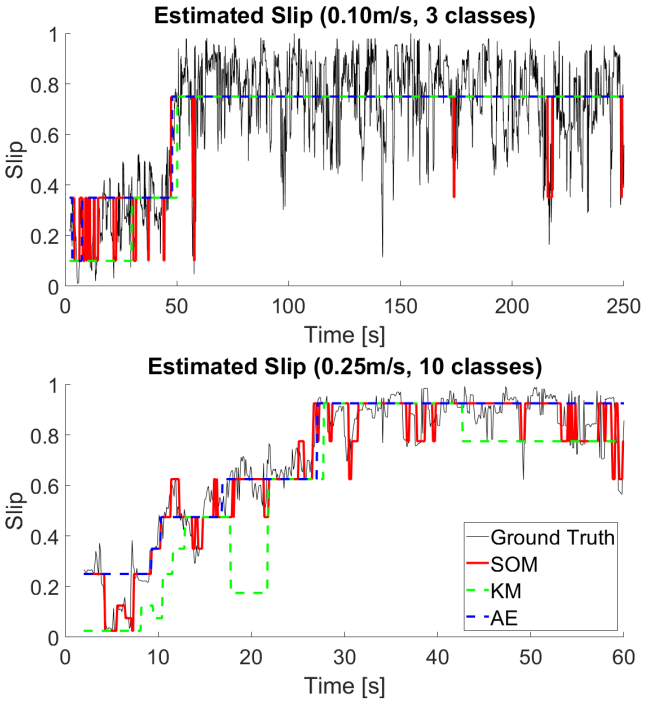


Figure 5. Estimated slip vs ground truth slip.

for each algorithm. When looking at correct responses, the SOM algorithm is most optimal for low and high slip, while k-means is optimal for medium slip. When looking at incorrect responses, SOM is very likely to incorrectly classify medium slip as high slip. This is not entirely negative; from a safety perspective, overestimating slip is acceptable and would result in more cautious (but slower) driving. KM is similarly likely to classify medium slip as high slip. However, it is by far the least accurate algorithm when classifying low-slip cases. The autoencoder is least accurate for medium slip cases, and again tends to overestimate slip.

Slight changes to class boundaries have a strong impact. For the [0, 0.3, 0.6, 1] case, each algorithm observes a 4-6% accuracy loss. This is due to a 20-30% increase in mis-classification of medium slip points as high slip points. Setting the division between medium and high slip at $s = 0.6$ does not provide as strong a delineation of the medium/high slip transition region. Class divisions should thus be chosen intelligently, with input from how well rover sensors can distinguish between certain slip levels.

Algorithm Tuning

Figure 7 presents differences in accuracy between the worst and best sets of parameters for each unsupervised algorithm. The autoencoder is generally the most sensitive, with accuracy losses of up to 25%. SOM is least sensitive, except for the case of ten slip classes. Note that while k-means can observe large variations between trials (Figure 4), it does not observe as much improvement on average.

Care should be taken to choose parameters that ensure optimal performance. For each algorithm, the most important parameters were found to be:

- **SOM**

1. Map size M (up to 13% improvement). Larger maps are useful for lower velocities and/or more classes, allowing

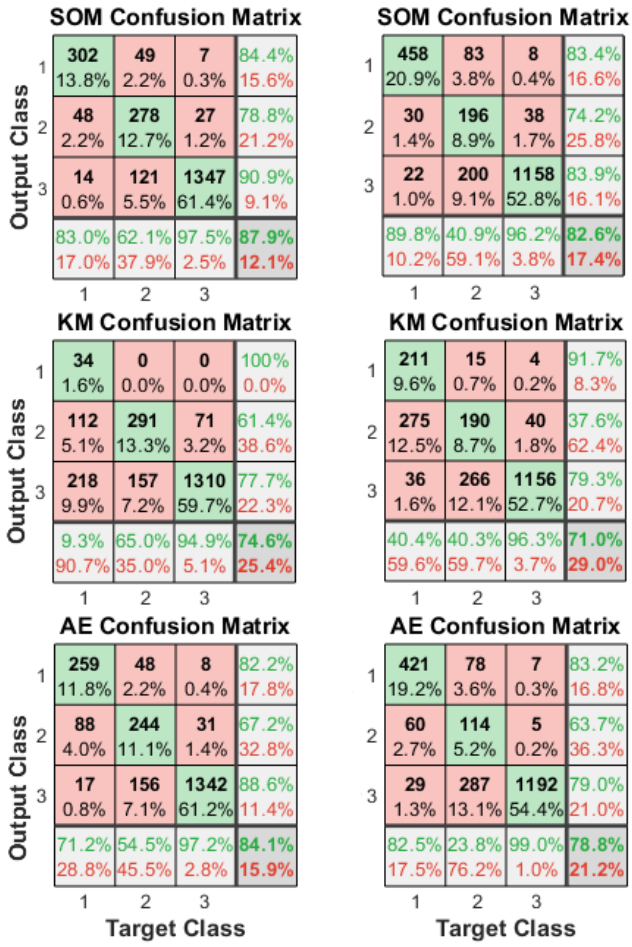


Figure 6. Confusion matrices for 0.10 m/s and 3 classes. The left three matrices use class boundaries [0, 0.2, 0.5, 1], while the right three matrices use [0, 0.3, 0.6, 1].

finer delineation of nearby slip clusters. Maps between 20x20 and 30x30 are recommended. Note that computational requirements increase according to M^2.

2. Choice of inputs (up to 8% improvement).
3. Size of neuron class filter F_{nc} (up to 4% improvement). $F_{nc} = 5$ avoids both outliers and excessive homogeneity.

- **KM**

1. Choice of inputs (up to 18% improvement).
2. Choice of distance function (up to 15% improvement). Either L1- or L2-norm functions are recommended.
3. Length of input filter F_{in} (up to 4% improvement). Taking a longer period of sensor data into account generally improves accuracy, though also introduces a delay. A length between 1-3s is recommended.

- **AE**

1. Number of neurons n_h (up to 12% improvement). Additional neurons are useful for lower velocities and/or more classes, allowing finer delineation of data features. 50-100 neurons in each layer is recommended. Note that computational requirements increase linearly with n_h.
2. Choice of transfer function (up to 10% improvement). A linear transfer function is recommended.
3. Choice of loss function (up to 5% improvement). Mean squared error is recommended.
4. Sparsity proportion S_p (up to 5% improvement). There

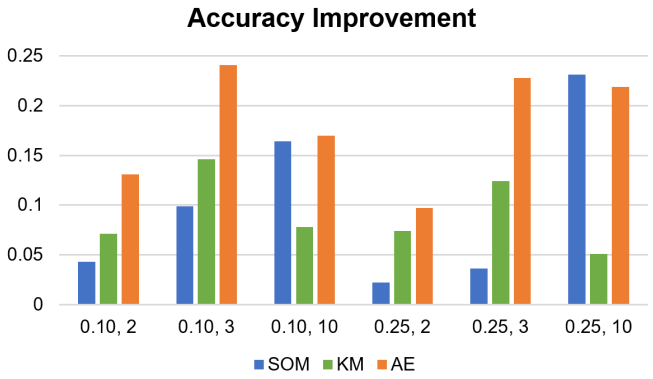


Figure 7. Accuracy improvement of each algorithm for different [velocity, # of classes], using the worst vs. best parameters that were found.

are many more neurons than input dimensions, and more sparsity (e.g. $S_p = 0.01$) ensures that neurons specialize by only activating for a small number of training inputs.

In general, we desire the rover to stop as soon as possible after detecting consistently high slip, which places a premium on faster response times and filters. Accuracy did not significantly increase with filters longer than 2s.

5. Conclusions

This study extends recent investigations into using machine learning to estimate wheel slip of a planetary exploration rover, by combining unsupervised learning with proprioceptive sensing. Three unsupervised algorithms – self-organizing maps, k-means clustering, and autoencoding – are trained to classify wheel slip into discrete classes. This presents several advantages when compared to many previous approaches: the algorithms are independent of terrain features or lighting, do not require estimation of vehicle or terrain parameters, are applicable across a range of velocities and inclinations, and rely only on commonly-available sensors. The algorithms themselves are generic and can independently train to detect slip for different terrain, environmental and vehicular conditions. The use of proprioceptive data ensures less computational complexity than VO-based methods.

During forward driving along a sandy incline between 0.10-0.25 m/s, self-organizing maps demonstrated the best accuracy in the two-class embedding detection case (93-97%) and the three-class case (87-93%). At higher velocities, 71% accuracy is displayed for the ten-class case (rising to 93% if adjacent classes are considered correct). From an accuracy perspective, SOM is optimal, regardless of velocity or the number of slip classes. It is also the most consistent over repeated training runs. It did, however, require the most processing time and memory, making it less suited to computationally-limited rovers. K-means is generally not recommended. Although it was by far the fastest algorithm, it faces a significant 5-30% accuracy loss compared to SOM. Autoencoders lie at a midpoint in performance, with accuracy 2-10% worse than SOM. Nevertheless the flexibility with which such networks can be structured and layered may allow further improvements.

Choice of sensor inputs remains crucial. A combination of IMU, encoder and motor current measurements proved effective, though the best input set differed between algo-

rithms and was not necessarily dependent on maximum slip correlation. Utilizing mean readings from multiple sensors is preferred for more consistent slip detection. However, classification ability is affected by many related factors, such as sensor placement on the vehicle; the structure, material, and weight distribution of the chassis; and terrain and environment parameters, which were not investigated in detail here.

The primary limitation on the results stems from the field trial data, which consisted of ten forward drives along sandy terrain and up a slope. This is a very narrow test case and ideally, the data would cover more terrain types over a longer driving period, to provide more examples of how low slip and high slip manifest. It is perhaps realistic to train a rover on limited data – since this may be necessary at the start of a real mission – but for assessing algorithm behavior in different terrains and the overall usefulness of the ML approach, more test data would be valuable.

Furthermore, the algorithms (as set up in this paper) use inputs from all wheels simultaneously. They thus cannot tell which individual wheels are slipping or how wheels are differently affected. It must also be noted that while each algorithm is unsupervised, some supervision is needed to associate SOM/KM clusters or AE features with slip classes. However, a relatively short period of labelled data is enough to enable classification – much less than what a fully-supervised algorithm would require.

Future work will explore some of these questions, implementing unsupervised ML across larger data sets and determining its accuracy when more varied conditions and scenarios are taken into account. Additional algorithms such as deep belief networks will be considered, as well as additional sensor types – for example, lower-noise sensors to improve slip estimation at low velocities. More investigation is also required into the processing and storage requirements of unsupervised algorithms, given the computationally-limited nature of rovers, with comparison to other prominent methods such as VO. Ultimately, unsupervised ML methods will be implemented on planetary rover hardware to further assess their suitability for robust, real-time slip estimation.

ACKNOWLEDGMENTS

The authors wish to acknowledge ProtoInnovations, LLC and MIT for assisting with the field trial data analyzed in this report.

REFERENCES

[1] NASA Jet Propulsion Laboratory, "Spirit Updates," 2009. [Online]. Available: mars.jpl.nasa.gov/mer/mission/status_spiritAll_2009.html, Accessed on: Sep. 18, 2018.

[2] NASA Jet Propulsion Laboratory, "Opportunity Updates," 2005. [Online]. Available: mars.nasa.gov/mer/mission/status_opportunityAll_2005.html, Accessed on: Sep. 18, 2018.

[3] NASA Jet Propulsion Laboratory, "NASA's Curiosity Rover Adjusts Route Up Martian Mountain," 2015. [Online]. Available: jpl.nasa.gov/news/news.php?feature=4596, Accessed on: Sep. 18, 2018.

[4] NASA Jet Propulsion Laboratory, "Breaks Observed in Rover Wheel Treads," 2017. [Online]. Available: jpl.nasa.gov/news/news.php?feature=6785, Accessed on: Sep. 18, 2018.

[5] J. Y. Wong, *Theory of Ground Vehicles*, 4th ed. Hoboken, NJ, USA: Wiley, 2008.

[6] K. Iagnemma and S. Dubowsky, *Mobile Robots in Rough Terrain. Estimation, Motion Planning, and Control with Application to Planetary Rovers*. Berlin, Germany: Springer, 2004.

[7] A. Angelova, L. Matthies, D. Helmick and P. Perona, "Learning and prediction of slip from visual information," *J. Field Robotics*, vol. 24, no. 3, pp. 205-231, Mar. 2007.

[8] R. Gonzalez and K. Iagnemma, "Slippage estimation and compensation for planetary exploration rovers. State of the art and future challenges," *J. Field Robotics*, vol. 35, no. 4, pp. 564-577, Jun. 2018.

[9] R. Ambrose *et al.*, "NASA Technology Roadmaps - TA4: Robotics and Autonomous Systems," NASA, Washington D.C., 2015.

[10] K. Iagnemma and C. C. Ward, "Classification-based wheel slip detection and detector fusion for mobile robots on outdoor terrain," *Auton. Robots*, vol. 26, no. 1, pp. 33-46, Jan. 2009.

[11] C. C. Ward and K. Iagnemma, "A dynamic-model-based wheel slip detector for mobile robots on outdoor terrain," *IEEE Trans. Robot.*, vol. 24, no. 4, pp. 821-831, Jul. 2008.

[12] E. Lakdawalla, "Curiosity Update, Sols 671–696: Out of the Landing Ellipse, into Ripples and Pointy Rocks," 2014. [Online]. Available: planetary.org/blogs/emilylakdawalla/2014/07241401-curiosity-update-sols-671-696.html, Accessed on: Sep. 18, 2018.

[13] G. Reina, L. Ojeda, A. Milella and J. Borenstein, "Wheel slippage and sinkage detection for planetary rovers," *IEEE/ASME Trans. Mechatronics*, vol. 11, no. 2, pp. 185-195, Apr. 2006.

[14] L. Matthies *et al.*, "Computer vision on Mars," *Int. J. Comput. Vis.*, vol. 75, no. 1, pp. 67-92, Oct. 2007.

[15] G. Reina, G. Ishigami, K. Nagatani and K. Yoshida, "Odometry correction using visual slip angle estimation for planetary exploration rovers," *Adv. Robot.*, vol. 24, no. 3, pp. 359-385, Mar. 2010.

[16] M. Maimone, Y. Cheng and L. Matthies, "Two years of visual odometry on the Mars Exploration Rovers," *J. Field Robot.*, vol. 24, no. 3, pp. 169-186, Mar. 2007.

[17] R. Gonzalez, D. Apostolopoulos and K. Iagnemma, "Slippage and immobilization detection for planetary exploration rovers via machine learning and proprioceptive sensing," *J. Field Robot.*, vol. 35, no. 2, pp. 231-247, Mar. 2018.

[18] M-R. Bouguelia, R. Gonzalez, K. Iagnemma and S. Byttner, "Unsupervised classification of slip events for planetary exploration rovers," *J. Terramechan.*, vol. 73, no. 10, pp. 95-106, Oct. 2017.

[19] R. Gonzalez, M. Fiacchini and K. Iagnemma, "Slippage prediction for off-road mobile robots via machine learning regression and proprioceptive sensing," *Robot. Auton. Syst.*, vol. 105, no. 7, pp. 85-93, Jul. 2018.

[20] R. Gonzalez and K. Iagnemma, "Soil embedding avoidance for planetary exploration rovers," *The 13th ISTVS European Conf.*, Rome, Italy, 2016.

[21] T. Kohonen, "Essentials of the self-organizing map," *Neural Networks*, vol. 37, no. 1, pp. 52-65, Jan. 2013.

[22] K. Obermayer and T. Sejnowski, *Self-Organizing Map Formation: Foundations of Neural Computation*. Cambridge, MA, USA: MIT Press, 2001.

[23] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

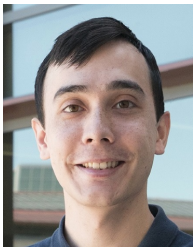
[24] B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: a strategy employed by V1," *Vision Research*, vol. 37, no. 23, pp. 3311-3325, Dec. 1997.

[25] C. M. Bishop, *Pattern Recognition and Machine Learning*. Berlin, Germany: Springer, 2006.

[26] *Advanced algorithms and controls for superior robotic all-terrain mobility*, ProtoInnovations LLC and MIT, Pittsburgh, PA, 2018.

[27] D. N. Reshef *et al.*, "Detecting novel associations in large datasets," *Science*, vol. 334, no. 6062, pp. 1518-1524, Dec. 2011.

BIOGRAPHY



Justin Kruger *received B.S. degrees in physics and mechatronics engineering from the University of Western Australia in 2016, and is studying an M.S. degree in aerospace engineering at Stanford University. He attended the NASA Ames International Internship program in 2018. His current research activities are focused on enhanced spacecraft autonomy and formation flying.*

Arno Rogg *received a M.S. in microengineering from the Swiss Institute of Technology of Lausanne (EPFL) in 2016. He is currently working at NASA Ames Research Center in the Intelligent Robotics Group. His work focuses on planetary rover research & development with a focus on rover mobility in difficult terrains and high reliability systems integration.*

Ramon Gonzalez *is the founder and CEO of robonity a worldwide tech startup. He has performed postdoctoral research in the Robotic Mobility Group at the Massachusetts Institute of Technology (3 years), and has been a visiting researcher at the Autonomous Systems Lab (ETH Zurich, Switzerland) and the University of Seville (Spain). He holds a Computer Science Engineering degree and a Ph.D. in mobile robotics from the University of Almeria (Spain). He has also worked at the University of Zaragoza (Spain) as a PhD Assistant. He is author of the monograph Autonomous Tracked Mobile Robots in Planar Off-Road Conditions (Springer, 2014). His main research interests include: modelling, localization and motion control of mobile robots and autonomous vehicles in outdoor conditions; terrain classification and characterization; computer vision and AI; and geostatistics applied to mobile robotics. He has received several awards including the Gold Medal of Andalucia (2017). Ramon currently serves on the editorial board of the Journal of Terramechanics.*