Efficient Generation of Accurate Mobility Maps Using Machine Learning Algorithms

Paramsothy Jayakumar
U.S. Army Tank Automotive Research,
Development & Engineering Center (TARDEC)
Warren, MI, USA

Dave Mechergui
U.S. Army Tank Automotive Research,
Development & Engineering Center (TARDEC)
Warren, MI, USA

Abstract

U.S Army's mission is to develop, integrate, and sustain the right technology solutions for all manned and unmanned ground vehicles, and mobility is a key requirement for all ground vehicles. Mobility focuses on ground vehicles' capabilities that enable them to be deployable worldwide, operationally mobile in all environments, and protected from symmetrical and asymmetrical threats. In order for military ground vehicles to operate in any combat zone, the planners require a mobility map that gives the maximum predicted speeds on these off-road terrains. In the past, empirical and semi-empirical techniques [1-2] were used to predict vehicle mobility on off-road terrains such as the NATO Reference Mobility Model (NRMM). Because of its empirical nature, the NRMM method cannot be extrapolated to new vehicle designs containing advanced technologies, nor can it be applied to lightweight robotic vehicles.

The mobility map is a function of different parameters such as terrain topology and profile, soil type (mud, snow, sand, etc.), vegetation, obstacles, weather conditions, and vehicle type and characteristics. The mobility map is obtained by discretizing a terrain map into different grids, each grid represents a certain geographical area of the terrain having its topology, soil type, and characteristics, and by super imposing a mobility measure. The mobility measure used is the speed-made-good which is an estimated maximum vehicle speed at a steady-state for each terrain grid. The grid colors are a function of these vehicle speeds. The obtained mobility map is a colored map representing the mobility measure, and the map can be labeled as 'go/no-go.' The 'go' color represents the geographical areas of the off-road terrain where the vehicle can move, and the 'no-go' represents the geographical areas of the terrain where the vehicle is immobile.

A physics-based method such as the discrete element method (DEM) [3] was identified by the NATO Next Generation NRMM Team as a potential high fidelity method to model the soil. The DEM method models the contact between the vehicle and soil as well as between soil particles, in which the normal and tangential contact forces are modeled. This method allows the capture of the soil deformation as well as its non-linear behavior. Hence it allows the simulation of the vehicle on any off-road terrain and have an accurate mobility map generated. The drawback of the DEM method is the required simulation time in addition to a lack of comprehensive validation data. It takes several weeks to generate the mobility map because of the large number of soil particles (millions) even while utilizing high performance computing.

One approach to reduce the computational time is to use machine learning algorithms to predict the mobility map. Machine learning [4-6] can lead to very accurate mobility predictions over a wide range of terrains. Machine learning is divided into two categories: the supervised and the unsupervised learning. Supervised learning requires the training data to be labeled into predetermined classes, while the unsupervised learning does not require the training data to be labeled. Machine learning can help generate mobility maps using trained models created from a minimum number of simulation runs. In this study different supervised

machine learning algorithms such as the support vector machine (SVM), the nearest neighbor classifier (*k*-NN), decision trees, and boosting methods were used to create trained models labeled as 2 classes for the 'go/no-go' map, 5 classes for the 5-speed map, and 7 classes for the 7-speed map. The trained models were created from the physics-based simulation runs of a nominal wheeled vehicle traversing on a cohesive soil.

1. Introduction

In this study we used machine learning to predict the mobility maps of a military wheeled vehicle crossing soft soil modeled as DEM particles. The vehicle mobility is function of the soil characteristics and several soil parameters are investigated: the soil cone index and soil friction. From paper [7], the following characteristics had a major effect on vehicle mobility: soil cohesion, soil friction, soil particle size, and soil density. Machine learning is a subfield of computer science and is closely related to computational statistics. The main difference is that machine learning is used to make future predictions from previous data set, whereas in statistics the results are inferred from the properties of the data set. Machine learning was developed for the purpose of pattern recognition and is now used in different fields such as speech recognition, in medicine, in search engines, finance, etc. It can be applied for autonomous vehicles that learn how to move on unknown terrains from their own experience. Machine learning can also be used to treat patients using medical records stored in data set to learn how future patients will respond to which treatments.

Machine learning is divided into two categories; the supervised machine learning and the unsupervised machine learning. In the supervised machine learning, a trained model is created from the data, where the response is labeled into different classes. The trained model is then used to predict the behavior of new data by classifying the response values of the new data. The unsupervised machine learning does not use classification instead it needs a large number of data set to predict the behavior of the new data set. The supervised machine learning contains different algorithms such as the support vector machines (SVMs), the k-nearest neighbor (k-NN), decision trees, and neural networks (NN). The unsupervised learning contains clustering, self-organized maps and hidden Markov models. A brief introduction to machine learning algorithms of the type supervised is given below.

1.1. Support Vector Machines

Wen et al [8] have studied the SVMs, in their work they mentioned that SVMs can tell the difference between two classes by learning from examples of these classes. SVMs estimates the optimal boundary that separates these two classes using a pair of parallel hyperplanes. If the training examples of these two classes are linearly separable, then the hyperplanes are the pair that separates these two classes with a maximum gap. The shortest distance from the separating hyperplane to the closest example is called the margin of the separating hyperplane. The training example is represented by a pair (xi,yi), where $xi \in \mathbb{R}^n$, $yi \in \{-1,1\}$, this for $i = 1, \dots m$. xi is the vector describing the ith example, yi describes the label of the class it belongs to. Let's take w a vector defining the normal direction of a certain hyperplane, any two parallel hyperplanes can be represented by the equation below.

$$wT\ x + b = \pm 1, \text{ where } b \in \mathbb{R} \quad (1)$$

We can find a pair of hyperplanes that gives the maximum margin by minimizing $\|w^2\|$ and satisfy the constraints in the equations above.

If the classes are non-separable, positive slack variables ξ_i, $i = 1,\dots, l$ are introduced in the constraints

$$x_i * w + b \geq +1 - \xi_i \quad \text{for } y_i = +1 \quad (2)$$

$$x_i * w + b \leq -1 + \xi_i \quad \text{for } y_i = -1 \tag{3}$$

To assign extra cost for errors, we have to change the objective of the function to be minimized from $\frac{\|w\|^2}{2}$ to

$$\frac{\|w\|^2}{2} + C\left(\sum_i \xi_i\right)^k \tag{4}$$

Where C is a parameter to be chosen, a big C value corresponds to a high penalty error.
For datasets with different classes having a nonlinear separation boundary, kernels are used as a mapping trick to make the data linearly separable. The used kernels are polynomial, Gaussian or radial basis functions. If the data is not linearly separable due to some minor nonlinearities present in the separation boundary, soft SVMs is used. Slack variables are introduced they account for the nonlinearities with respect to the separation boundary.

1.2 *K*-Nearest-Neighbor (*K*-NN)

K-NN is a simple and effective non-parametric method used for classification and regression [9]. To apply k-NN we need to choose an appropriate k value, the classification outcome is a function of this value. In order to have a good classification we need to run the simulations with different k values and chose the value that gives the best classification. *k*-NN studies the labels of neighboring data points to predict the classes or labels of a new data set. Figure 1 shows the k-NN classification for two classes, the classes are separated by a boundary. The classification is done based on distance metrics, the distance metric between the neighboring points from the same class is minimized. The distance between points from different classes is maximized. There are a few distance metrics used in the k-NN algorithm among them is the Euclidian, Chebyshev, Minkowski, and Canberra. The major drawbacks of the k-NN are the low efficiency: a lazy learning method that is dependent on the value of k. Wang [10] proposed a method that examines multiple sets of the nearest neighbors rather than just one set of k-NN. The proposed method is based on contextual probability. The idea is to aggregate the support of multiple sets of nearest neighbors for different classes, this gives a more support value. The various metrics to determine the distance are defined below:

Euclidean distance: $$\Gamma = \sqrt{(\mathbf{p}-\mathbf{q})^T(\mathbf{p}-\mathbf{q})} \tag{5}$$

Minkowski distance: $$\Gamma = \left(\sum_{i=1}^{n} |p_i - q_i|^p\right)^{\frac{1}{p}} \tag{6}$$

Canberra distance: $$\Gamma = \sum_{i=1}^{n} \frac{|p_i - q_i|}{|p_i| + |q_i|} \tag{7}$$

Chebyshev distance: $$\Gamma = \lim_{k\to\infty}\left(\sum_{i=1}^{n} |p_i - q_i|^k\right)^{\frac{1}{k}} \tag{8}$$

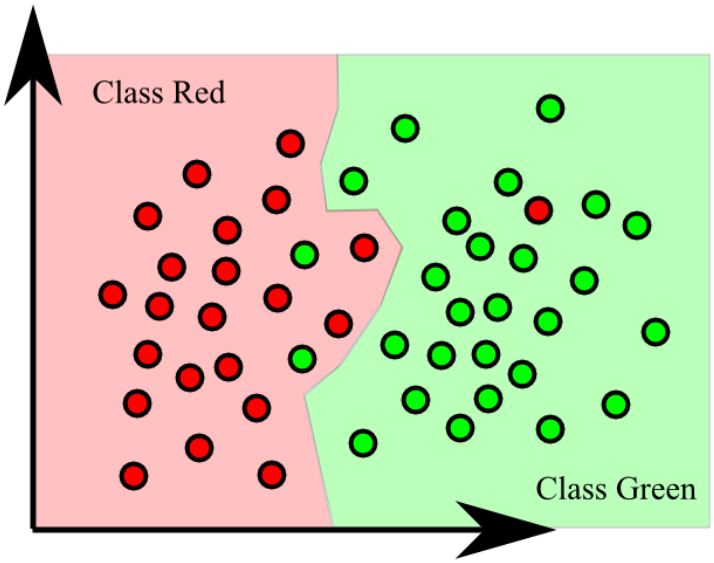


Figure1: k-NN classification

1.3 Decision Trees

Decision tree learning is a method for approximating discrete-valued target functions, the learned function is represented by a decision tree [11]. Decision trees are applied to a range of tasks from learning to diagnose medical cases. Decision trees classify things by sorting them down. The trees contain nodes and branches, each node represents an attribute and each branch contains the attribute value.

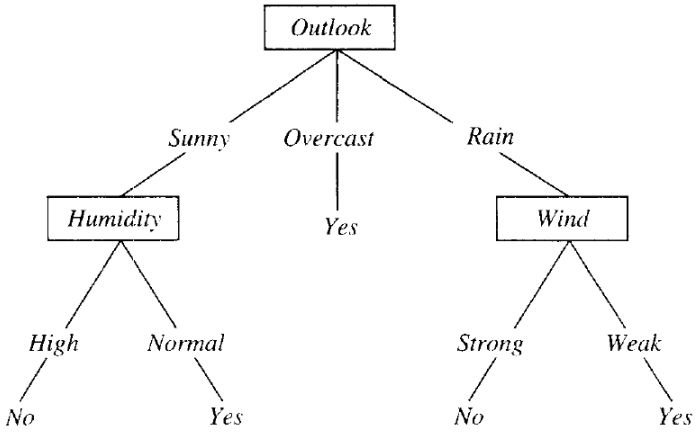


Figure 2: Decision tree classification

Decision trees method is suited for problems with:

- Instances are represented by attribute-value pairs
- The target functions have discrete output values
- The training data contain errors; the decision tree learning methods are robust to errors: the errors in classification of the training models and errors in the attribute values.

- When the training data contain missing attribute values; decision trees can be used when some training examples have unknown values.

Because of this decision trees have found a wide range of applications such as learning to classify patients by their disease, loan applicants by their defaulting payments, equipment by their malfunctions. Decision tree algorithms can have difficulties when there is a noise in the data, or when the number of training data is too small to produce a representative sample of the true target function. In these cases the algorithm can lead to an over fit [12].

1.4 Neural Networks

Artificial neural networks (ANN) are computing systems inspired by biological neural networks that constitute the brain [13]. The ANN is a framework for different machine learning algorithms to process complex data. They are composed of processing units, neurons, and weighed connections between these neurons. The neurons co-operate to perform a desired function. ANN are used in a wide range of areas such speech recognition, machine translation, social network, and medical diagnosis. Neural networks are used for classification: pattern recognition, feature extraction, and image matching. They are also used in prediction. Neural networks have the ability to learn; they figure out how to perform their function on their own, they determine their function based only on inputs [14]. The output of a neuron is a function of the weighted sum of the inputs plus a bias (figure 3). The function of the entire neural network is a computation of the outputs of all the neurons. The neural networks structure is composed of input and output layers and hidden layers (figure 4). Weights in neural networks are the most important factor in determining its function. In supervised trainings, the neural network is supplied with the inputs and the desired outputs, the response of the network to the inputs is measured. The weights are modified to reduce the difference between the actual and desired outputs. In unsupervised trainings, the neural network is supplied only with the inputs. The networks adjusts its own weights so that similar inputs cause similar outputs. The network identifies the patterns and differences in the inputs without any external assistance.

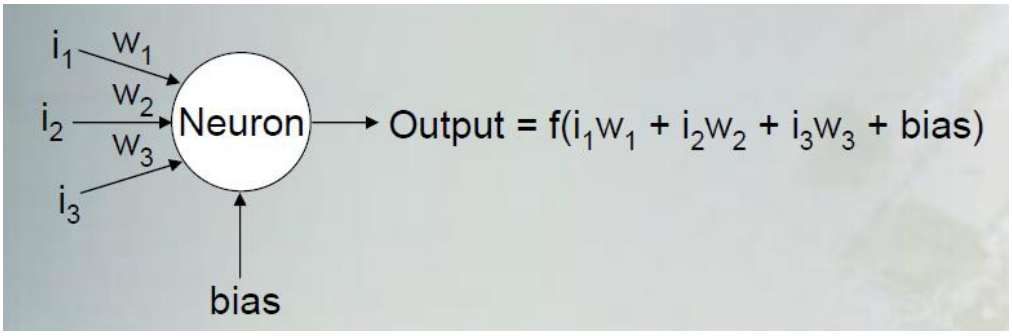


Figure 3: Structure of a neuron

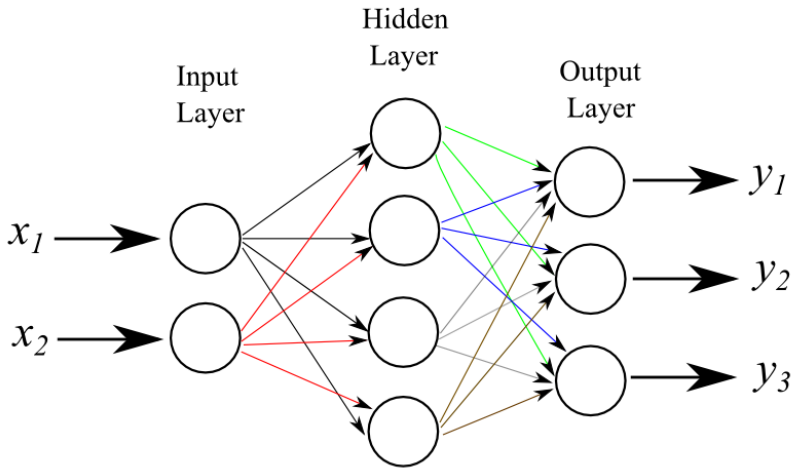


Figure 4: Neural networks structure

1.5 Ensemble Methods

The goal of the ensemble learning methods is to create an ensemble of individual classifiers, which is accurate. The accuracy of the ensemble is greater than that of the individual classifiers. The two famous techniques for building ensembles are boostrap aggregation or "bagging" [15] and the Adaboost family of algorithms, "boosting" [16]. Both these techniques operate by taking a basic learning algorithm and use it with different training sets. In bagging each training set is created from a copy of the original training set. The Adaboost algorithm maintains a set of weights of the original training set. The weight is increased for the examples that are misclassified, and it is decreased for the examples that are correctly classified. Bagging and to a lesser extent boosting are viewed as methods to solve an algorithm's instability to improve classification accuracy [17]. An algorithm is unstable when small change in the training model causes a large changes in the learned classifier.

1.6 Kriging

Kriging is formulated to be used as an interpolation method. Kriging methods use a polynomial model that fits to a response surface. The local departures from the global fit are estimated using semi-variogram models [18]. The sampled points are interpolated to give a resulting approximation, the kriging is represented by a certain function that can estimate values. This function is described in [19], and is represented by the expression below:

$$\bar{f}(x_o) = \sum_{l=0}^{r} \lambda\, l_l\, p_l(x_0) + Z(x_0) \qquad (9)$$

Where $p_l(x_0)$ is the $l-th$ order term in a polynomial basis of maximum order r and λ_l is the least-squares solution to the set of normal equations $f(x_j) = \lambda_l p_l(x_j)$, $j = 1,2,\ldots,N$. In equation (9), $Z(x_0)$ is the realization of the Gaussian process with zero mean, $E[Z(x_j)] = 0$ and a covariance,

$E[Z(x_j)Z(x_q)] = \sigma^2 R_{jq}, j,q = 1,2,..N$ where σ^2 is the process variance, Rjq is the correlation model of the process, and R_{jq} is chosen to be of the form.

$$R = R_{jq} = R(\theta, x_j, x_q) = \prod_{k=1}^{n} \gamma k(\theta_k, d_k), \quad (11)$$

θ, the shape parameter, $d_k = (x_{kj} - x_{kq}), k = 1,2\ldots,n$.

1.7 Latin Hypercube Sampling

Design optimization requires a large number of simulation runs, which can be time consuming. Advances in computer science had reduced the simulation time and allowed the use of more design variables that can be simulated at a reasonable amount of time. One way to reduce the computational time is the use of surrogate models known meta-models [20-25]. The design optimization of the surrogate models identifies locations in the design space where the simulations will be conducted [26, 27]. The response data are collected at these locations, more surrogate models are fitted to the data. One or more models are selected to calculate the responses at points in the design space for which the actual responses are not measured yet. The quality of fit that defines the performance of the surrogate model during optimization depends on the design of experiments (point location and density) [28, 29]. The quantity of fit is the discrepancy between response predicted by the surrogate model and the response.

A design of experiments with n_p points and n_v variables is written as $n_p \times n_v$ matrix

$X = \left[x_1, x_2, \ldots, x_{np}\right]^T$, where each row $x_i = \left[x_{i1}, x_{i2}, ..., x_{in}\right]$ represents a sample and each column represents a variable. The Latin hypercube represents the following advantages:

1. The number of samples (points) is not fixed
2. The sampling points are orthogonal
3. The sampling points do not depend on the surrogate model.

A Latin Hypercube Design with n_p points is constructed in a way that each is divided into n_p equal levels. So there is only one point (sample) at each level, the points in the design space are located randomly. The LHD process fills the design space randomly, so the space can be poorly filled. The optimization of the space filling quality of the Latin hypercube sampling design is a challenging problem. One way to solve the space filling is to consider a n_v dimensional sphere around each design point. The larger the radius of the sphere, the better the space filling property of the design provided that the sphere does not cross the boundary of the design space. Optimization of the Latin hypercube design to produce uniform distribution of the sampling points in design space is obtained by maximizing the sphere radius.

2. Vehicle Model

The vehicle multibody dynamics model is shown in figure 5. It is made of 33 rigid bodies: main chassis; 4 wheels; 4 upper suspension control arms; 4 lower suspension control arms; 4 knuckles; 4 drive shaft; 2 tie rods; and steering rack. The bodies are connected using spherical, revolute, prismatic and CV joints. A gear differential model [30] is used to model the front and rear differentials. A rotational actuator is used to model the drive torque at the drive shaft (tractive torque after the transmission system). The torque is limited to be between the two curves shown on Figure 6. The curves represent the WOT (wide-open throttle) torque (blue curve) and the maximum engine brake torque (purple curve). The x axis represents the drive shaft angular velocity. Four rotational actuators at the wheels are used for modeling the brakes. The maximum brake torque as a function of the wheel angular velocity is shown in figure 7. The total sprung mass of the vehicle is 4,430 kg. The mass of one wheel is 50 kg. Each tire polygonal surface consists of

about 6600 triangles (Figure 5). The tire diameter is 0.97 m. The tires' surfaces are set as slave contact surfaces for the DEM particles. The tires are modeled as rigid bodies.

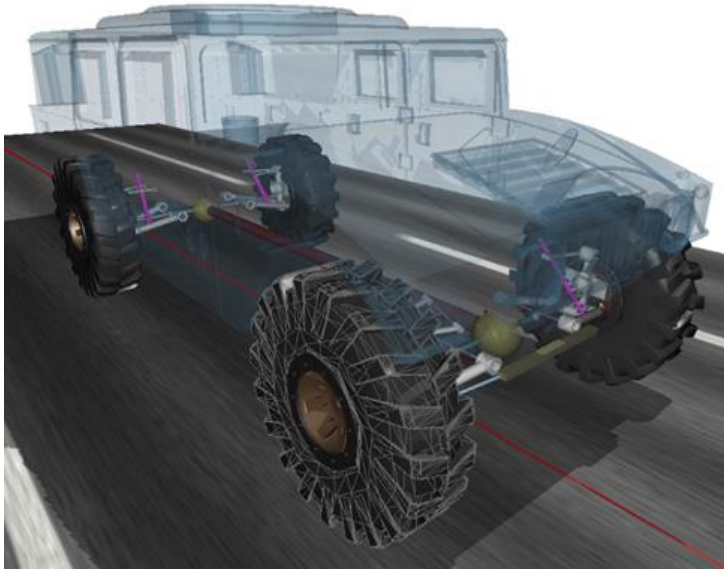


Figure 5. Multibody dynamics model of a HMMWV

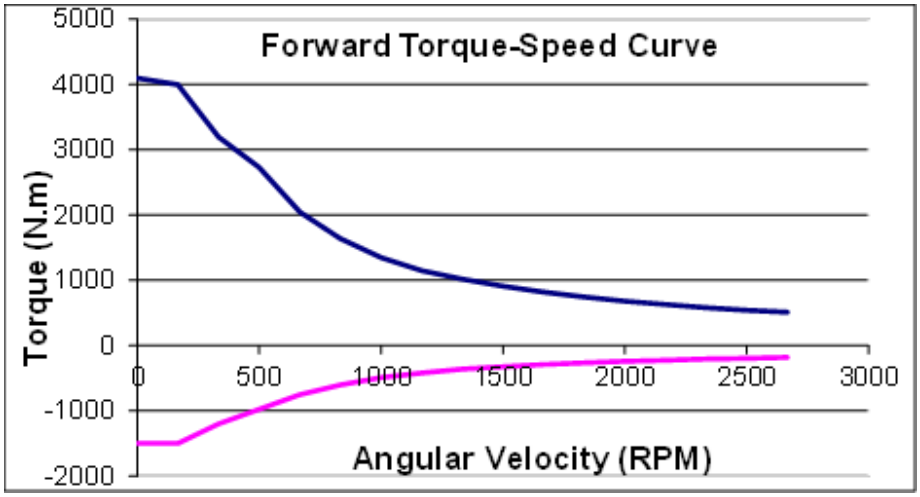


Figure 6. WOT engine torque (blue) and maximum engine brake torque (purple)

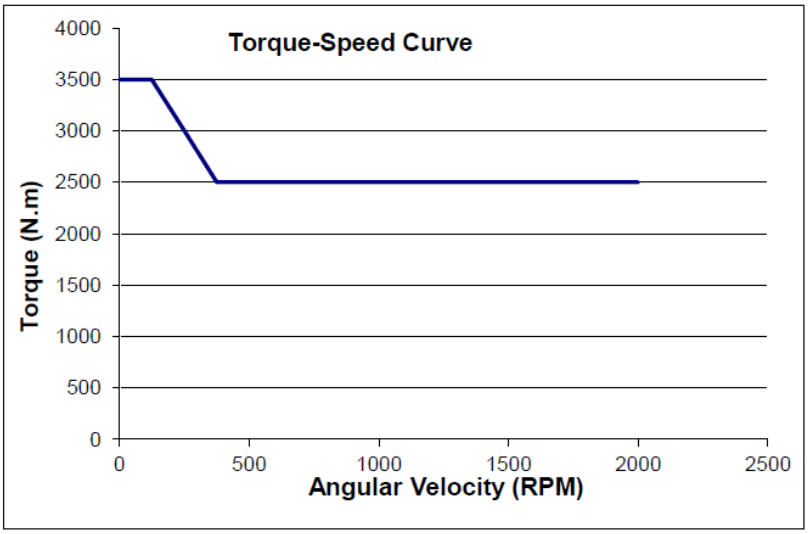


Figure 7. Maximum brake torque at each wheel.

3. DEM Soil

The DEM soil material properties are listed in Table 2. The DEM model has 620,000 point-type particles, the particle diameter is 30 mm (the total number of particles varies with particle diameter). The soil particles are inside a box that is 9.3 m long, 3.48 m wide, and 0.9 m high with an open top (Figure 8). The soil is compressed in the vertical direction using a flat lid for one second using a pressure of 33.3 kPa. The lid is removed after the consolidated soil settles to a height of about 0.4 m. A Cartesian search grid for speeding up inter-particle contact search with a resolution of 310 (length), 116 (width) and 30 (height) is used [30]. The side length of a grid cell was chosen to be almost equal to the diameter of a particle.

A moving soil patch technique is used in which particles which are far behind the vehicle are continuously eliminated then reemitted as new particles in front of the vehicle [30]. This is achieved using a rectangular particle emitter, a leveling cylinder and plate, and a bounding sphere (see Figure 8). When a particle goes outside the bounding sphere it is deactivated and then immediately reemitted as a new particle from the rectangular particle emitter from a random point on the surface of the emitter. Once the x-coordinate of the c.g. of the vehicle reaches the initial x-coordinate of the center of the bounding sphere, then the bounding sphere center is moved with the c.g. of the vehicle frame using a script function along with a prescribed motion constraint.

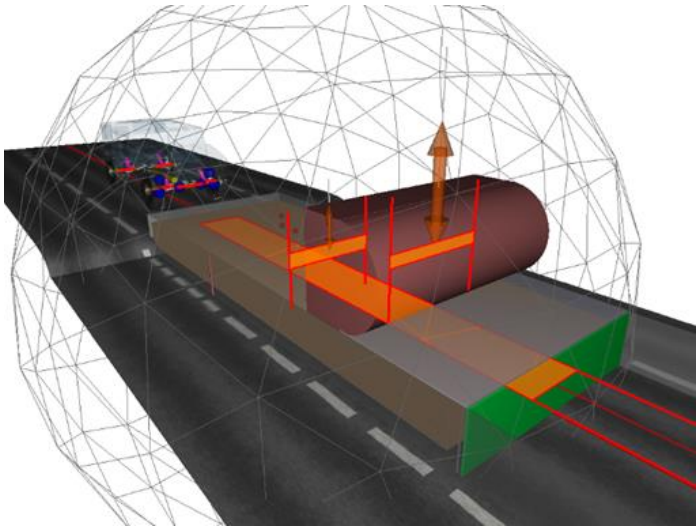


Figure 8. Moving soil box model consisting of a moving box, a leveling lid, a leveling cylinder/plate (shown in red), and a rectangular particle emitter (shown in green).

The moving soil patch technique allows to reduce the number of DEM particles for long vehicle travel distances, so the simulation can complete in a 7 days. The vehicle needs at least a 400 m long patch to reach the speed of 60 mph from rest and run a few seconds at steady-state. If the patch width is 3.5 meters, compressed soil depth is 0.4 meters, and compressed particle diameter is 26.5 mm, then the required number of particles is about 67,000 particles per meter of terrain. So for a 400 m long terrain patch the number of particles is 27 millions. At current simulation computational speeds, a 35 sec simulation with 27 million particles will take about 9 months to complete. However, for the simulations in this paper where the moving terrain patch is 9.3 m long and the number of particles is about 620,000, a 35 sec vehicle simulation takes about 7 days on a 32 core HPC node.

The DEM soil was characterized using the cone penetrometer simulations. The effect of DEM inter-particle cohesion and friction coefficients on the soil strength using the cone index (CI) was studied [32]. The cohesion coefficient was varied between 0 and 12 and the friction coefficient was varied from 0.05 to 0.4. The simulation results are shown in figure 9.

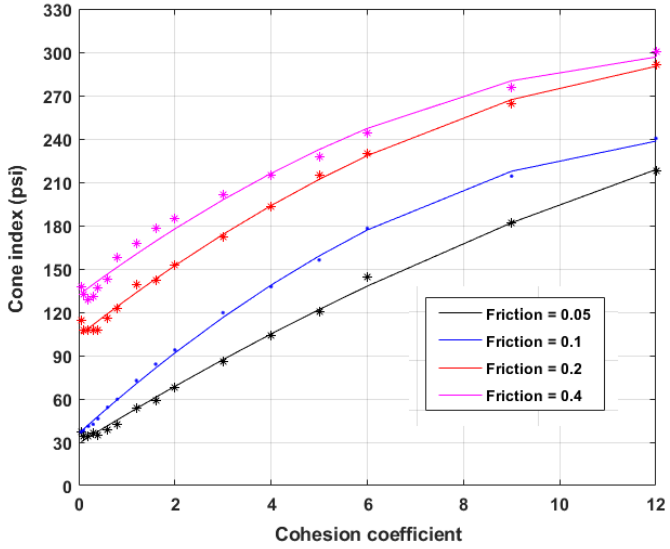


Figure 9. CI versus cohesion factor (*f*) and friction (μ).

Table 1. Parameters of the cone penetrometer

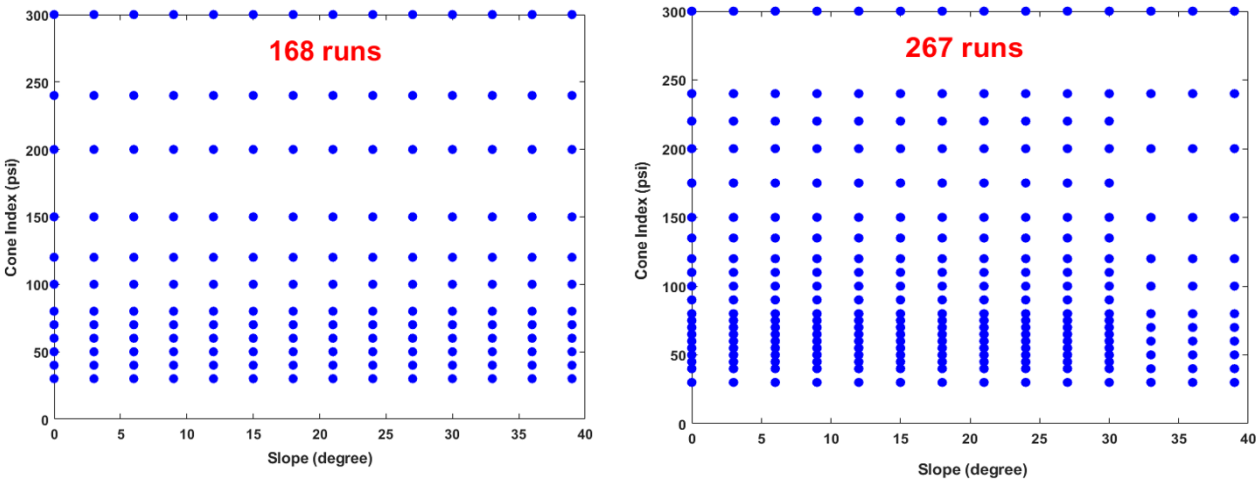
Cylindrical container diameter	2 m
Consolidating lid pressure	33.3 kPa
Cone penetrometer base diameter	0.375 m
Cone penetrometer length	0.7 m
Cone penetrometer cone angle	30°
Penetrometer speed	0.1 m/s

Table 2. Parameters of the DEM soil material model

Unstressed (unconsolidated) particle diameter	0.03 m
Particle mass density	1800 kg/m^3
Inter-particle friction coefficient	0.1
Particle to tire/cone penetrometer friction coefficient	0.5
Inter-particle viscosity	0
Inter-particle damping per unit area	2.1×10^4 N/m^3/s
Particle stiffness (slope of repulsion stress versus penetration strain in Figure 7)	4.42×10^7 N/m^2
Plastic strain versus compressive stress	[33]
Nominal maximum adhesion stress versus plastic strain curve	[33]
Plastic relaxation speed	0.045 m/s

4. Model Identification and algorithm selection

Different machine learning algorithms were investigated to determine the best algorithm that can be used to generate accurate mobility maps. Two soil parameters were varied; the cone index and the soil longitudinal slope. The cone index was varied between 30 and 300 psi, and the terrain slope varied between 0 and 30 degrees. No sampling pattern was used in this study. The main criteria is the vehicle mobility; the vehicle does not move when the slope is greater than 30 degrees. Beyond a cone index value of 240 psi the terrain becomes hard, so the study will focus on the area comprised between a cone index less than 240 psi and a slope less than 30 degrees. In order to select the algorithm we varied the number of simulation runs between 100 and 528. Training models were generated from these data, this is for different algorithms.



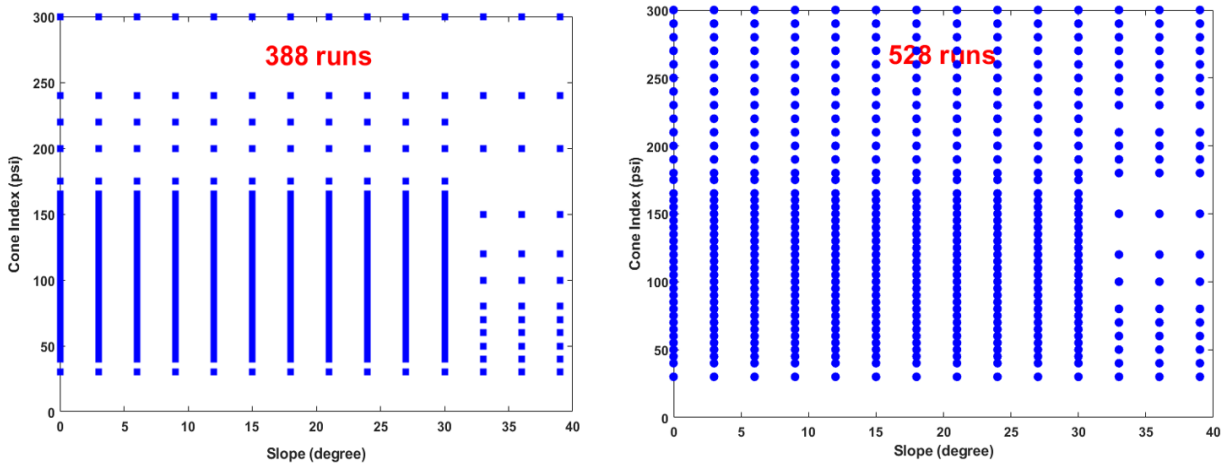


Figure 10: Soil cone index and terrain slope distribution used as training data

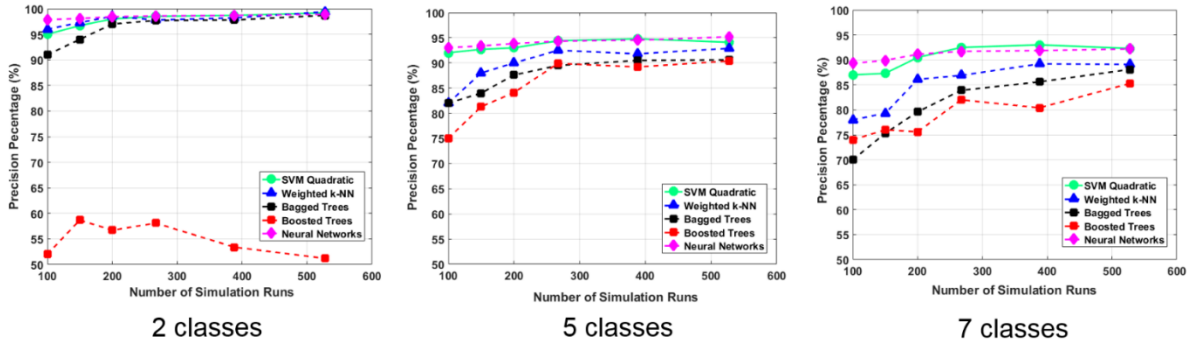


Figure 11: Machine learning algorithms

From figure 11 the following can be deduced: the accuracy of the training model does not change after a certain number of training runs for the different classes studied. The accuracy of the training models decreased when the number of classes increases, this can be explained by an increase of the generalized error. The SVM Quadratic and the Neural Networks training models are more accurate than the training models of the other algorithms. Especially for the two classes the accuracy of the training models (SVM Quadratic and Neural Networks) is higher than 99%.

5. Mobility map prediction

Training models were created from 388 simulation data points with two design variables: cone index and terrain slope. The response is the vehicle speed which is measured at the steady state. Mobility maps were generated for two, five, and seven classes using the SVM and Neural networks training models.

5.1 Mobility map prediction for two classes

Figure 12 shows the mobility map (go no-go) of the trained data as a function of cone index and terrain slope. The red circles mean that the vehicle has a zero speed and the green circles mean the vehicle has a speed greater than zero. The trained machine learning models are used to predict mobility maps for a terrain with a given set of cone indices and terrain slopes. The slope was varied from 0 to 39 degrees at a 0.5 degree increment. The cone index was varied from 0 to 300 psi at an increment of 6 psi. The SVM trained model results in the mobility map in Figure 13 and the NN trained model results in the mobility map in Figure 14. The predicted mobility map generated by the SVM algorithm is very similar to the map of the trained data, which demonstrates the prediction accuracy of this algorithm.

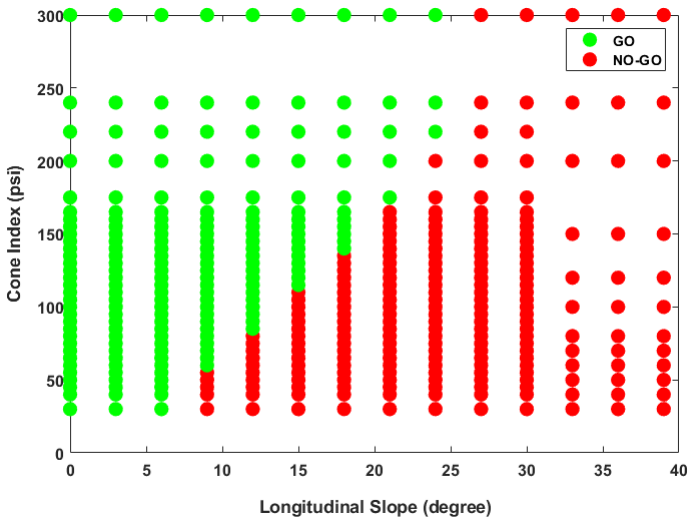


Figure 12: 2-class mobility map of the training data (388 runs)

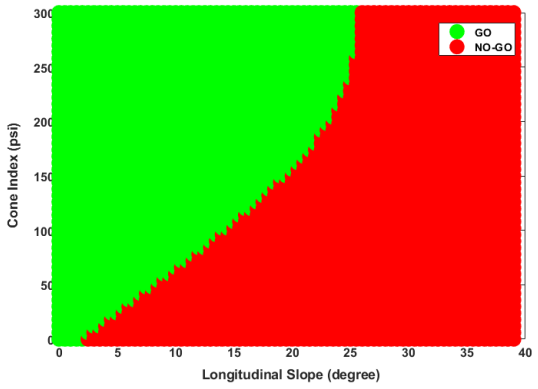


Figure 13: Predicted map for two classes using SVM

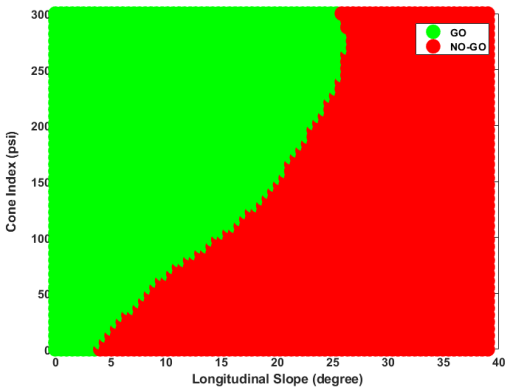


Figure 14: Predicted map for two classes using NN

5.2 Mobility map prediction for five and seven classes

Like in Chapter 5.1 two machine learning algorithms were used: the SVM and the NN to predict a 5-class and 7-class mobility maps, the variables are the cone index and the terrain slope. Figures 16-19 show the different speed classes of the predicted mobility maps. The SVM and the NN algorithms give maps that show some differences. The mobility maps in Figures 16 and 17 are different from the map represented in the training data (Figure 15(a)). The mobility maps in Figures 18 and 19 are also different from the map represented in the training data (Figure 15-b). This can be explained by the fact that when the number of classes increases the separating boundary becomes nonlinear. As a consequence the generalization error increases and the prediction error also increases. This was shown in Chapter 4, the prediction error increased from 1% for a two-class trained model to 5% for a 5-class model. This is true for all the studied algorithms; SVM, Neural networks, bagged and boosted ensembles.

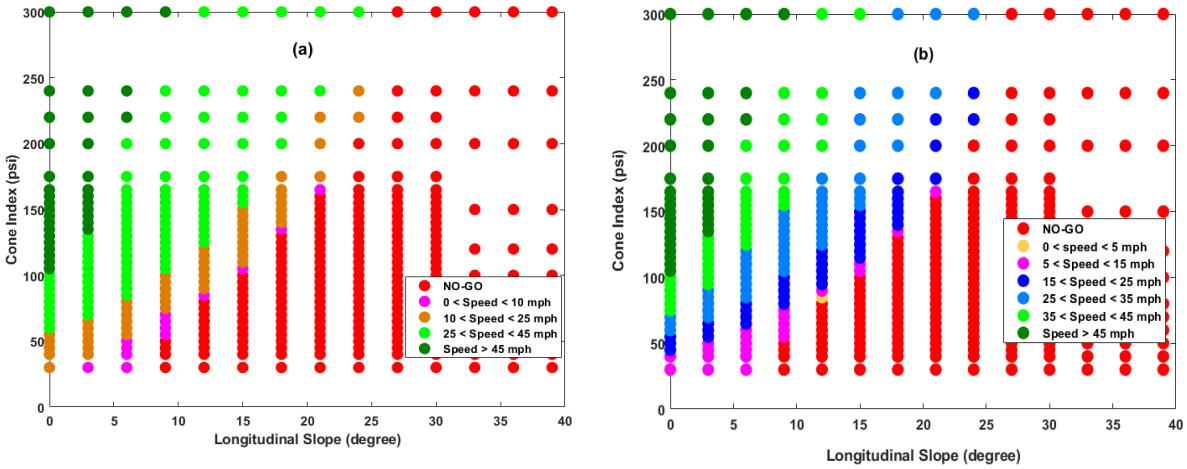


Figure 15: 5-class (a) and 7-class (b) mobility maps of the training data (388 runs)

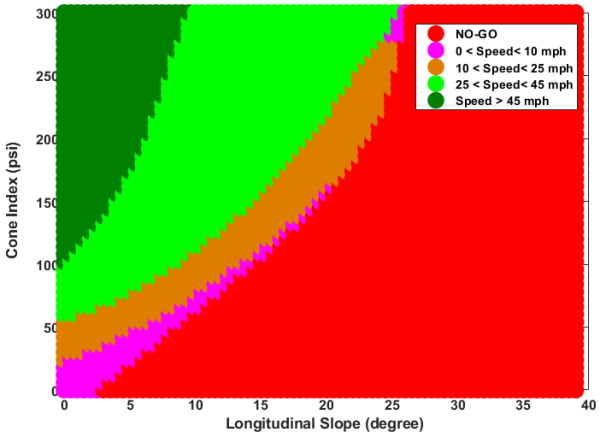


Figure 16: Predicted map for five classes using SVM

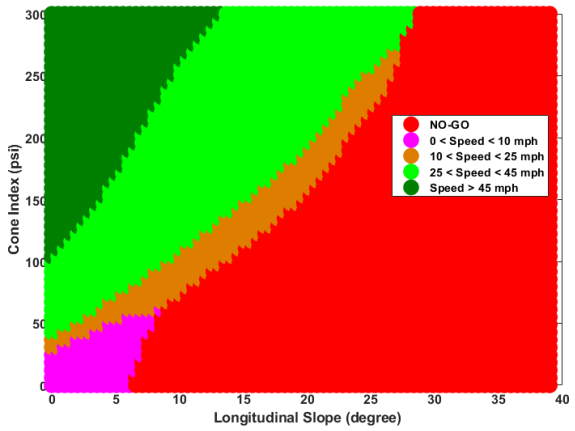


Figure 17: Predicted map for five classes using NN

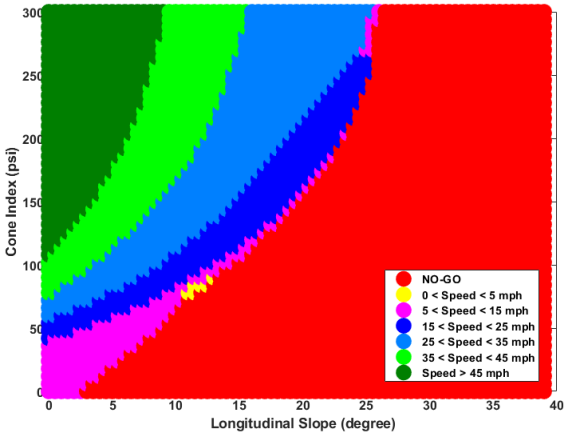


Figure 18: Predicted map for 7 classes using SVM

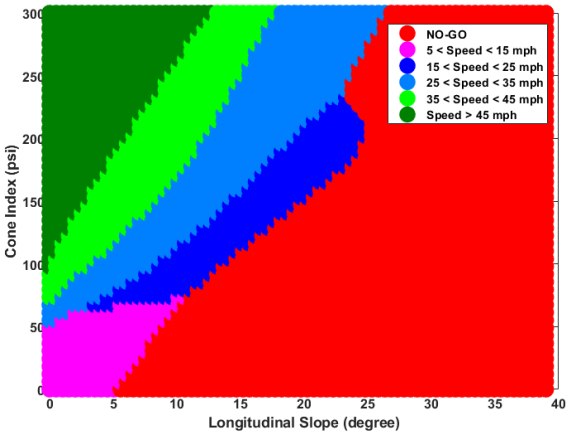


Figure 19: Predicted map for 7 classes using NN

5.3 Validation error using machine learning

Trained models that were created from 388 simulation data were used to assess the validation error; this is for the SVM and the Neural networks methods. The simulation data is set of two variables: the cone index and the terrain longitudinal slope, the response is the vehicle speed. The validation data are also a set of simulation data (206 runs). These data have also cone indices and terrain slopes as variables and the speed as a response. The validation data were obtained using the Latin Hypercube Sampling technique to have a random sampling. Figure 20 shows the distribution of the new dataset, they are shown in green squares, whereas the trained data are shown in blue squares. The validation data are distributed as follows: the cone index varies between 160 and 300 psi, and the terrain slope varies between 0 and 39 degrees. Figures 21 and 22 show the validation errors for the two algorithms, this for different classes, Table 3 also shows this. From this table we can deduce that the validation error of the Neural Networks model is smaller than that of the SVM for different classes. The validation error increases with the number of classes for both algorithms.

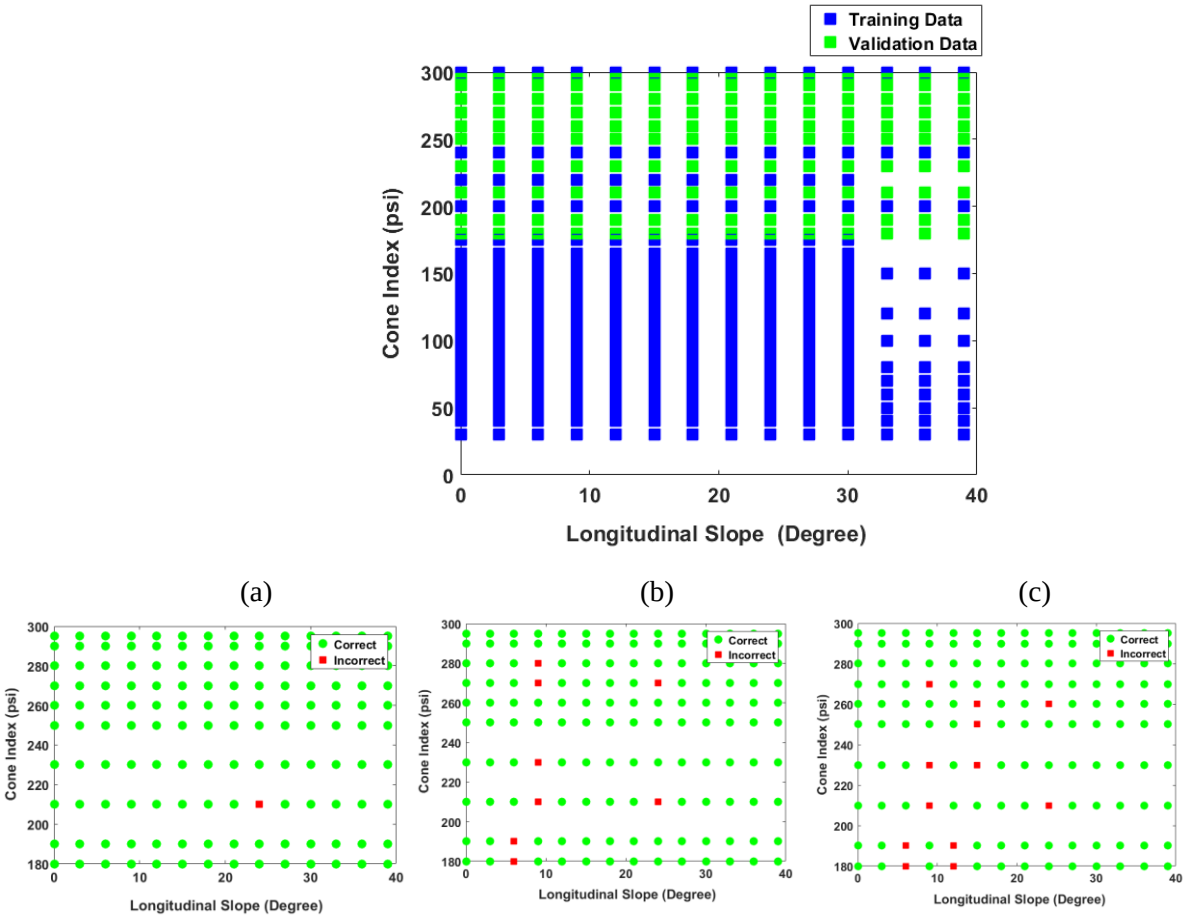


Figure 21: Validation error; (a) for two classes, (b) for five classes, (c) for seven classes using SVM

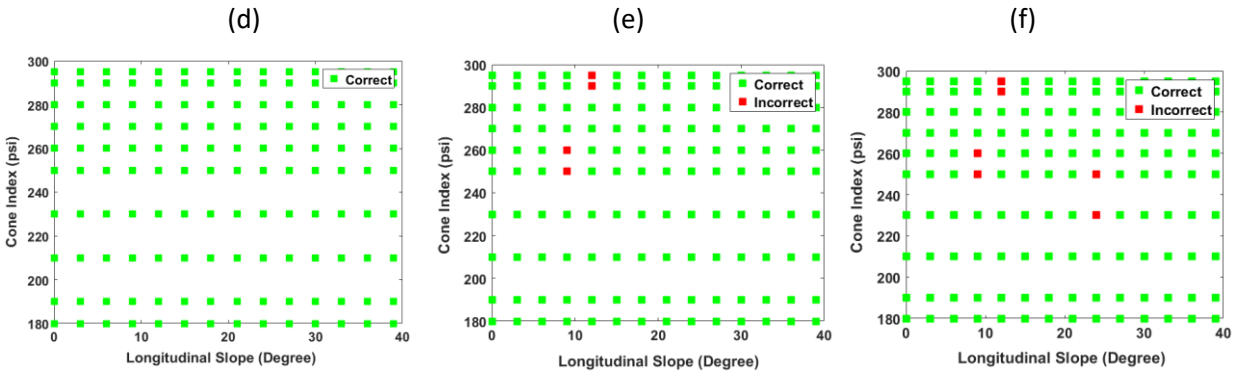


Figure 22: Validation error; (d) for two classes, (e) for five classes, (f) for seven classes using NN

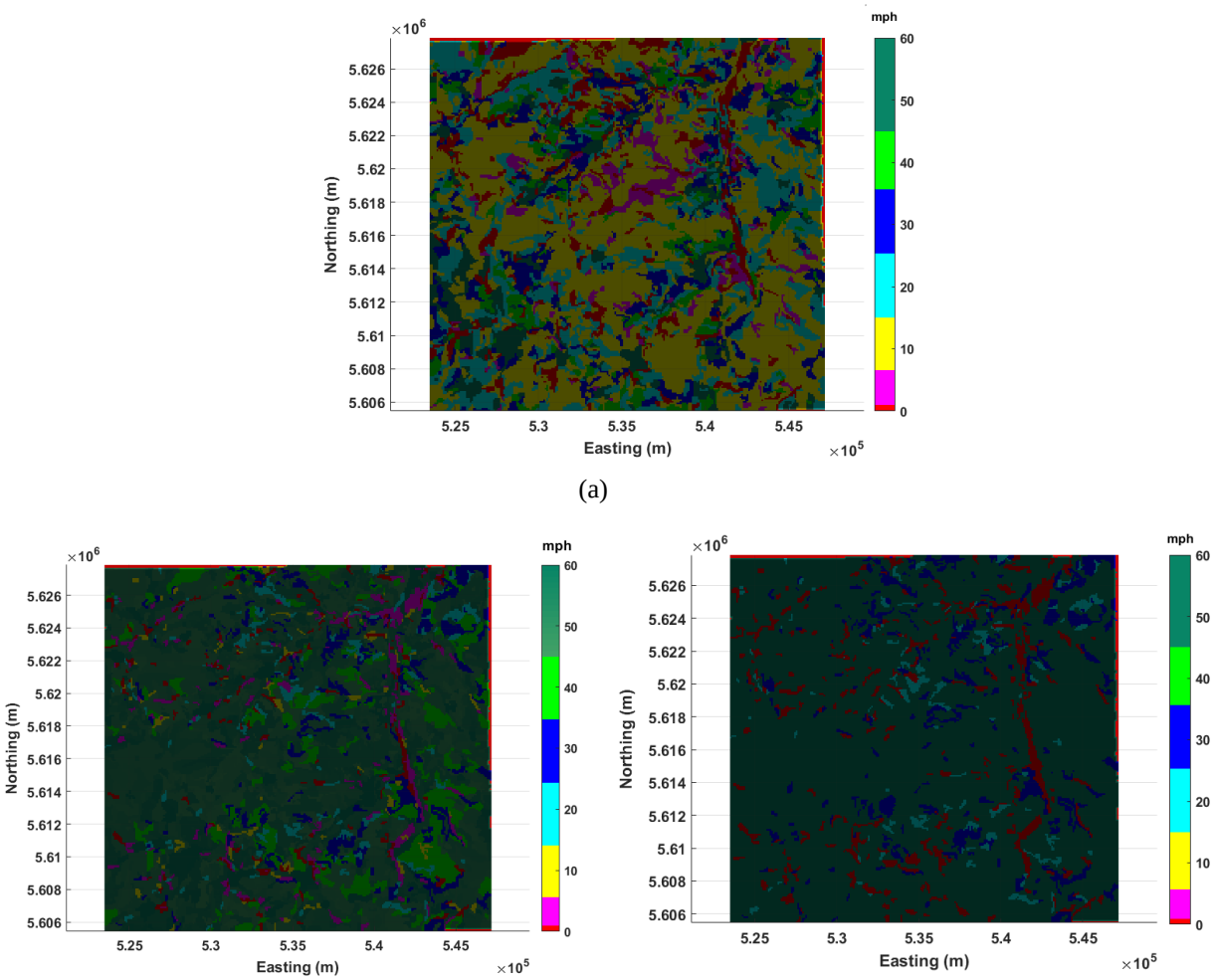
Table 3: Validation error vs algorithm and class number

Variable	Training model					
	SVM			Neural networks		
Number of classes	Two	Five	Seven	Two	Five	Seven
Incorrect predictions	1	8	12	0	4	6
Validation error (%)	0.50	3.98	5.82	0	1.95	2.90

6. Comparison of the Mobility Maps Generated by different Methods

In this section different mobility maps are shown below, these maps were generated by different techniques, the variables are the cone index and the terrain longitudinal slope. These values were taken from a real terrain, which contains 2707 units, each unit has its own topology, area, slope, cone index, soil type, roughness, …, etc. The maps represent different mobility classes (see legend), each speed class has its own color. The map in figure 23 (a) is obtained using the NRMM technique, which is an empirical method, the map was generated using Matlab. Figure 23 (b) represents the mobility map predicted by the Neural Networks, Matlab was also used to generate the map. Figure 23 (c) represents the mobility map predicted by the SVM, Matlab was used to generate the map. The maps look different, the overall speed in the NRMM map is lower than that in the maps generated using the SVM and the Neural Networks. This difference can be explained by the following:

- The actual DEM soil model does not contain all the parameters that are represented in the NRMM soil. In the actual DEM soil, the obstacles, the stems and the vegetation are not represented, these factors have an effect on the vehicle speed. The vehicle model with DEM soil unlike in the NRMM case does not use the brake to slow down when negotiating a turn or going around an obstacle.
- The actual DEM soil model might not accurately represent the true physical properties of the soil. The properties of DEM are modeled as function of the inter-particle friction and cohesion, there was no validation of the actual soil model. Some validation should be done in the future to correlate the DEM soil model to the actual soil.



(a)

(b)

(c)

Figure 23(a): NRMM map, Figure 23(b): Neural Networks map, Figure 23(c): SVM map

7. Conclusions

In this study the objective is to generate accurate mobility maps for vehicles running on off-road terrains. A nominal terrain was modeled as DEM soil, this soil model is physics-based. The DEM soil accurately represents the physical properties of the actual terrain. Using the simulations of a nominal vehicle on this nominal terrain one can generate the mobility map of this terrain. Because of the size of the terrain it would take months to finish the runs necessary to generate the map. One way to reduce the number of simulation runs and cut down the simulation time is to use machine learning. Machine learning can help by creating trained models from a small number of simulation runs. These trained models can be used to predict the mobility map of a given terrain. In order to predict a reliable mobility map, the trained model must be accurate enough.

In this study different machine learning algorithms were studied to predict the mobility maps, the number of simulation runs to create the algorithm was investigated. The most accurate algorithms are the SVM quadratic and the neural networks. The accuracy of the algorithm increased with the number of simulation data and reached a plateau, beyond a certain number the accuracy did not improve.

The accuracy of the algorithm decreased when the number of classes increased; this might be explained by the fact that the generalized error increased. When the different classes have a nonlinear separation boundary, polynomial kernels are used as a mapping trick to separate the different classes.

The mobility maps predicted by machine learning used only 15 % of the simulation runs, this allowed to save thousands of hours of simulation time.

The maps generated by NRMM and predicted by machine learning showed differences, the NRMM map showed lower speeds. This is due to the nature of the soils, unlike the NRMM soil model the DEM soil model does not contain stems, vegetation and obstacles. These factors can have an effect on the vehicle speed, they slow it down. The DEM soil is modeled as a moving patch, so the vehicle does not brake to negotiate turns and go around obstacles. Another explanation to the difference in mobility maps might be explained by the formulation of the DEM soil model. The soil model might not accurately represent the physical properties of the actual terrain. The DEM soil model was not validated against the actual terrain. Validation of the DEM soil in the future would help the machine learning trained models predict more accurate mobility maps.

References

[1] Ahlvin, R., and Haley, P., NATO Reference Mobility Model, Edition II, User's Guide, in Technical Report Number GL-92-19, U.S. Army Waterways Experiment Station, Corps of Engineers, Vicksburg, MS. 1992.

[2] Haley, P., Jurkat, M., and Brady, P. J., NATO Reference Mobility Model, Edition I, Users Guide, in Technical Report Number 12503, U.S. Army Tank-Automotive and Armaments Command, Warren, MI. 1979.

[3] Dasch, J., Jayakumar, P., Bradbury, M., Gonzalez, R., Hodges, H., Jain, A., Iagnemma, K., Letherwood, M., McCullough, M., Priddy, J., and Wojtysiak, B., *ET- 148 Next-Generation NATO Reference Mobility Model (NG-NRMM), in STO-TR-AVT-ET-148*, J. Dasch and P. Jayakumar, Editors. 2016, STO/NATO.

[4] Boutell, M. R., Luo, J., Shen, X., and Brown, C. M., Learning multi-label scene classification. Pattern Recognition, 37(9):1757-1771, 2004. ISSN 0031-3203.

[5] Burges, C. J. C., A Tutorial on Support Vector Machines for Pattern Recognition. Data Mining and Knowledge Discovery 2, 121-167, 1998.

[6] Barber, D., and Williams, C. K. I., 1997. 'Gaussian Processes for Bayesian Classification via Hybrid Monte Carlo'. In M. C. Mozer, M. I. Jordan, and T. Petsche (eds.), Advances in Neural Information Processing Systems, vol. 9, pp. 340. The MIT Press.

[7] Jayakumar, P., Mechergui, D., and Wasfy, T. M., Understanding the Effects of Soil Characteristics on Mobility. Proceedings of IDETC/CIE 2017. ASME 2017 International Design Engineering Technical Conferences. 13th International Conference on Multibody Systems, Nonlinear Dynamics, and Control (MSNDC). August 6 – 9, 2017, Cleveland, OH, USA.

[8] Wen, T., Edelman, A., and Gorsich, D., "A fast projected conjugate gradient algorithm for training support vector machines," Contemporary Mathematics: theory and applications, American Mathematical Society, pp. 245-263, Boston, MA, 2001.

[9] D. Hand, H. Mannila, P. Smyth: Principles of Data Mining. The MIT Press. (2001).

[10] Wang, H., Nearest Neighbours without k: A classification Formalism based on Probability, Technical Report, Faculty of Informatics, University of Ulster, N. Ireland, UK (2002).

[11] Mitchell, T., 1997, Machine Learning, First Edition, McGraw-Hill Education, New York City, USA.

[12] Schaefer, C., (1993). Overfitting avoidance as bias. Machine Learning, 10, 113-152.

[13] Gurney, K., An *Introduction to Neural Networks*, Taylor & Francis, 1997.

[14] Saravanan, K., and Sasithra, S., "Review on classification based on artificial neural networks", Int. J. Ambient Systems & Applications, Vol. 2(4), doi: 10.5121/ijasa.2014.2402, December, 2014.

[15] Breiman, L., (1996). Bagging predictors. Machine Learning, 24 (2), 123-140.

[16] Freund, Y., & Schapire, R. E., (1996). Experiments with a new boosting algorithm. In Proc. 13th International Conference on Machine Learning, pp. 146-148. Morgan Kaufmann.

[17] Breiman, L., (1994). Heuristics of instability and stabilization in model selection. Tech. rep. 416, Department of Statistics, University of California, Berkeley, CA.

[18] Li, H., Mulay, S. S., Meshless Methods and Their Numerical Properties, CRC Press, 2013.

[19] Oishik, S., Davis, S., Jacobs, G., and Udayakumar, H. S., Evaluation of convergence behavior of metamodeling techniques for bridging scales in multi-scale multimaterial simulation, Journal of Computational Physics, Vol. 294, pp. 585-604, 2015.

[20] Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, H. P., Design and Analysis of Computer Experiments, *Statistical Science*, Vol. 4 (4), pp. 409-435, 1989.

[21] Simpson, T. W., Peplinski, J. D., Koch, P. N., and Allen, J. K., Meta-models for computer based engineering design: Survey and recommendations. *Engineering with Computers*, Vol. 17, No. 2, pp. 129-150, 2001.

[22] Queipo, N. V., Haftka, R. T., Shyy, W., Goel, T., Vaidyanathan, R., and Tucker, P. K., Surrogate-based analysis and optimization, *Progress in Aerospace Sciences*, Vol. 41, pp. 1-28, 2005.

[23] Simpson, T. W., Toropov, V., Balabanov, V., and Viana, F. A. C., Design and Analysis of Computer Experiments in Multidisciplinary Design Optimization: a Review of How Far We Have Come – or Not, in: *Proceedings of the 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Victoria, Canada, September 10-12, 2008. AIAA 2008-5802.

[24] Kleijnen, J. P. C., *Design and analysis of simulation experiments*, Springer Verlag, 2007.

[25] Forrester, A. I. J., and Keane, A. J., Recent advances in surrogate-based optimization, *Progress in Aerospace Sciences*, Vol. 45, No. 1-3, pp. 50-79, 2009.

[26] Montgomery, D. C., *Design and analysis of experiments*, John Wiley & Sons, 2004.

[27] Simpson, T. W., Lin, D. K. J., and Chen, W., Sampling Strategies for Computer Experiments: Design and Analysis, *International Journal of Reliability and Applications*, Vol. 2, No. 3, pp. 209-240, 2001.

[28] Giunta, A. A., Wojtkiewicz, S. F., Eldred, M. S., Overview of modern design of experiments methods for computational simulations. *41st AIAA Aerospace Sciences Meeting and exhibit*, Reno, NV, AIAA-2003-0649, 6-9 January 2003.

[29] Goel, T., Haftka, R. T., Shyy, W., and Watson, L. T., Pitfalls of using a single criterion for selecting experimental designs, *International Journal for Numerical Methods in Engineering*, Vol. 75, No. 2, pp. 127-155, 2008.

[30] Wasfy, T.M., Wasfy, H. M., and Peters, J. M., High-Fidelity Multibody Dynamics Vehicle Model Coupled with a Cohesive Soil Discrete Element Model for Predicting Vehicle Mobility, in 11th ASME International Conference on Multibody Systems, Nonlinear Dynamics, and Control (MSNDC). 2015: Boston, MA.

[31] Ahmadi, S., Wasfy, T. M., Wasfy, H. M., and Peters, J. M., High-Fidelity Modeling of a Backhoe Digging Operation using an Explicit Multibody Dynamics Code with Integrated Discrete Particle Modeling Capability, in 2013 International Design Engineering Technical Conference, 9th International Conference on Multibody Systems, Nonlinear Dynamics, and Control (MSNDC). 2013, ASME.

[32] Wasfy, T.M., Jayakumar, P., and Mechergui, D., Understanding the Effect of Discrete Element Soil Model's Parameters on Ground Vehicle Mobility. To appear in the ASME, 2019.

[33] Wasfy, T.M., Jayakumar, P., Mechergui, D., and Sanikommu, S., Prediction of Vehicle Mobility on Large-Scale Soft-Soil Terrain Maps Using Physics-Based Simulation, International Journal of Vehicle Performance, Vol. 4, No. 4, pp. 347-381, 2018.