Novelty Detection and Online Learning for Vibration-based Terrain Classification

Christian WEISS [a,1] and Andreas ZELL [a]
[a] *Computer Science Department, University of Tübingen, Germany*

Abstract. In outdoor environments, a great variety of ground surfaces exists. To ensure safe navigation, a mobile robot should be able to identify the current terrain so that it can adapt its driving style. If the robot navigates in known environments, a terrain classification method can be trained on the expected terrain classes in advance. However, if the robot is to explore previously unseen areas, it may face terrain types that it has not been trained to recognize. In this paper, we present a vibration-based terrain classification system that uses novelty detection based on Gaussian mixture models to detect if the robot traverses an unknown terrain class. If the robot has collected a sufficient number of examples of the unknown class, the new terrain class is added to the classification model online. Our experiments show that the classification performance of the automatically learned model is only slightly worse than the performance of a classifier that knows all classes beforehand.

Keywords. vibration-based terrain classification

Introduction

In outdoor environments, a mobile robot faces many different types of ground surfaces. Each surface poses different hazards to the robot. One type of hazards are positive and negative obstacles, e.g. rocks or ditches. Other dangers originate from the properties of the ground surface itself, for example that it is very rough or slippery. Such dangers are called *non-geometric hazards* [1]. To avoid accidents or damage, a robot should be able to identify the current terrain in order to adapt its driving style. A common way to determine the terrain type is to group the terrain into classes that have known properties, e.g. grass or gravel. These classes are learned from training examples.

The most common methods for terrain classification use laser scanners (e.g. [2,3]) or cameras (e.g. [4,3,5]). A way to detect non-geometric hazards is vibration-based terrain classification, which was first suggested by Iagnemma and Dubowsky [6]. The main idea is based on the observation that traversing different terrain types induces different kinds of vibrations in the body or the wheels of the robot. Commonly, the vibrations are measured by accelerometers and the characteristic vibrations of different terrain classes are learned from training examples. A method based on probabilistic neural networks has been presented by DuPont et al. [7]. An approach based on linear discriminant analysis

[1]Corresponding Author: Christian Weiss, Computer Science Department, University of Tübingen, Sand 1, 72076 Tübingen, Germany; E-mail: c.weiss@uni-tuebingen.de.

was suggested by Brooks and Iagnemma [8]. We proposed a method that uses a support vector machine [9,10]. Stavens et al. focus on assessing the roughness of the terrain instead of grouping the ground surface into classes [11]. There also exist systems that combine vision and vibration sensing [12,13,14].

These approaches can only identify terrain classes that are contained in the training set. If the robot explores unknown areas, it is likely to encounter terrain it has never seen before. In this case, the new terrain will be classified wrongly as a class from the training set. Unlike the other approaches, the one presented by Brooks [8] can classify terrain as "unknown". This occurs, however, if a test example is equally similar to two terrain classes, and does not indicate that the current terrain is new. A possibility to detect new terrain is novelty detection which detects if a test sample is dissimilar to the training set (and therefore novel). Novelty detection approaches have been presented, for example, based on nearest-neighbor techniques [15], string matching [16], multi-layer perceptrons [17], one-class support vector machines (SVM) [18] or habituation [19]. An overview over different novelty detection methods can be found in [20,21].

In the paper at hand, we present a vibration-based terrain classification system that is able to detect if the robot traverses a ground surface it has not been trained to identify. We achieve this by novelty detection based on Gaussian mixture models (GMM) which have been sucessfully used for novelty detection in other domains, e.g. the detection of masses in mammograms [22] or sensor fault detection [23]. We chose GMMs because they are relatively simple and fast. In initial experiments, they also performed better than one-class SVMs. If in our system the robot has collected enough samples of a new class, this class is added to the model online. Our experimental results show that the classification performance of automatically learned models is only slightly worse than the performance of models that were trained manually on the same classes.

The rest of this paper is organized as follows. Section 1 gives an overview of our system and Section 2 describes the components in more detail. Section 3 presents our experimental results. Finally, Section 4 concludes the paper and suggests future work.

1. System Overview

The first step in our system is an offline training phase. As training data, we use vibration data of a number of terrain classes that are known to the robot right from the start. We represent vibration data by accelerations measured in three directions: perpendicularly to the ground floor (z-acceleration), left-to-right (y-acceleration) and front-to-back (x-acceleration) [10]. We split these x-, y- and z- vibration signals into smaller segments corresponding to a period of 1 s. At a measurement frequency of 100 Hz, this leads to three 1×100 vectors per terrain segment. We then transform each vector individually to the frequency domain using a log-scaled power spectral density (PSD), as suggested by Brooks [8]. After this, we concatenate the PSDs of the x-, y- and z- vibrations of a terrain segment to create the feature vector of the segment. Next, we normalize the feature vectors in the training set such that each feature has mean 0 and standard deviation 1. Additionally to such a feature vector, our system holds two other representations for a terrain segment. The first one is the feature vector reduced by principal component analysis (PCA) to dimension 1×20. The second one is formed by the raw acceleration data of the segment. In each stage of the system, the appropriate representation is used.

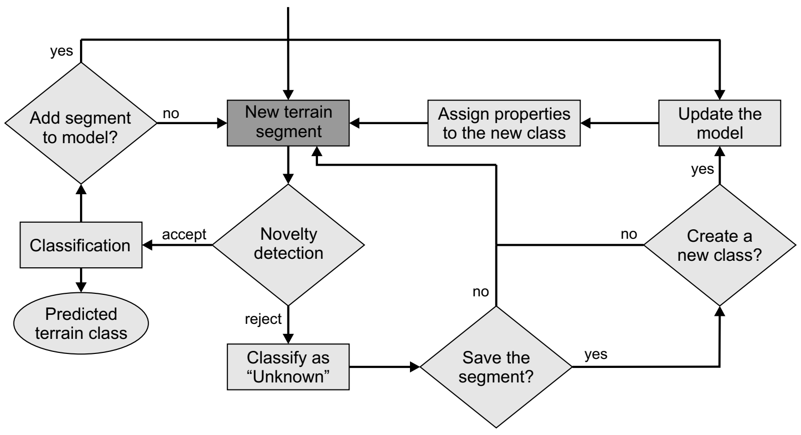


Figure 1. Overview of the classification phase of our system.

To complete the training phase, we train a k-nearest-neighbor (kNN) classifier on the initial training set. Additionally, we create an initial set of Gaussian mixture models for novelty detection.

The classification phase is depicted in Fig. 1. This phase forms a repeating cycle starting with the collection of a new one-second terrain segment and the computation of the corresponding feature vector $\mathbf{f}$. In the following step, the novelty detection method checks if $\mathbf{f}$ belongs to one of the known classes ($\mathbf{f}$ is "accepted") or if the class of $\mathbf{f}$ seems to be unknown so far ($\mathbf{f}$ is "rejected"). In case of acceptance, the kNN classifier predicts the segment's terrain type. Next, the system decides whether to add $\mathbf{f}$ to the training set. Being able to update existing classes is especially useful for increasing the number of training examples of newly created classes until the training set is balanced.

If the new terrain segment $\mathbf{f}$ is "rejected", the system decides if $\mathbf{f}$ should be stored in a buffer for further use, or if $\mathbf{f}$ is discarded. This step is an attempt to separate rejected outliers of known classes from truly unknown terrain segments. The next stage checks if the number of stored feature vectors in the buffer exceeds a given threshold. This indicates that enough examples of the unknown class are available to create a new class. In this case, the classifier must be retrained and new GMMs for novelty detection must be computed. Finally, the system tries to assign some properties like hardness or bumpiness to the new class.

2. System Components

2.1. Novelty Detection Using GMMs

The purpose of the novelty detection stage is to predict whether or not a newly aquired terrain segment $\mathbf{f}$ belongs to a class that is contained in the training set. For this purpose, we follow a novelty detection approach described in [24]. The main idea is to model the distribution of the available training data by a Gaussian mixture model. If a test example cannot be explained by the model, i.e. has a low probability given the model, it is likely to belong to a class that is not present in the training set. Our terrain classification system works on the feature vectors whose size have been reduced by PCA to dimension $1 \times d$.

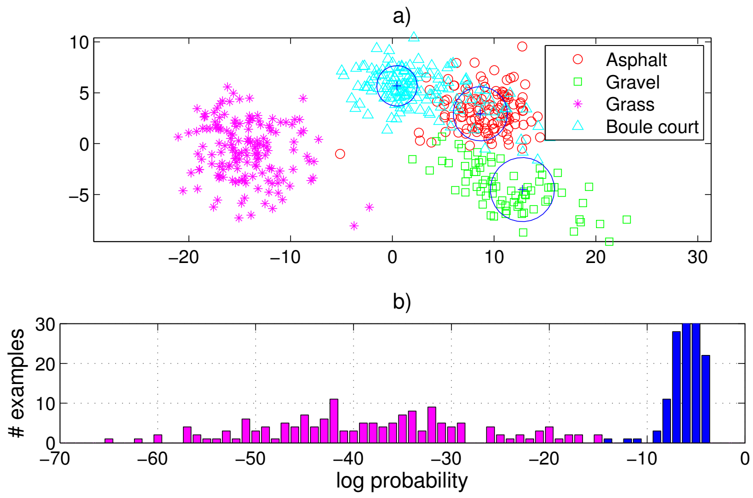


Figure 2. a) GMM with three spherical Gaussians fitted to vibration data (100 examples per class) of asphalt, gravel and boule court. Grass is used for testing. The centers of the Gaussians are indicated by a blue plus sign. Blue circles show one standard deviation error bars. The data dimension was reduced to 2 by applying PCA. b) Histogram of log probabilities on the training set (dark blue) and on the test set (light purple). Values larger than 30 were cut for clearer visibility.

In experiments, we found $d = 20$ to be a good trade-off between novelty detection performance and computation time. Smaller d generally led to decreasing novelty detection performance, while for larger d, generating GMMs becomes to slow.

In a GMM, a probability density function is expressed as a linear combination of Gaussian basis functions. If there are M Gaussians, the model is given by

$$p(\mathbf{x}) = \sum_{j=1}^{M} P(j)p(\mathbf{x}|j), \qquad (1)$$

where the *mixing coefficient* $P(j)$ can be viewed as the prior probability of the j-th component, and the $p(\mathbf{x}|j)$ are the Gaussian component density functions. By constraining the mixing coefficients by $\sum_{j=1}^{M} P(j) = 1$ and $0 \leq P(j) \leq 1 \forall j = 1, \ldots, M$ and having $\int p(\mathbf{x}|j)d\mathbf{x} = 1$, the model represents a probability density function.

Each of the Gaussian densities is represented by a mean vector μ_j of dimension d and a covariance matrix $\mathbf{\Sigma}_j$. We use spherical Gaussians with $\mathbf{\Sigma}_j = \sigma_j^2\mathbf{I}$, where $\mathbf{I}$ is the identity matrix. Fig. 2 a) shows an example of a GMM with three Gaussians fitted to vibration data. In practice, we use 16 Gaussians to form a GMM. For fitting a GMM to given data in an unsupervised fashion, one must determine the parameters of the GMM, i.e. the means and standard deviations of the Gaussians. This is done by maximizing the data likelihood or, equivalently, by minimizing the negative log likelihood $-\sum_{n=1}^{N} \log p(\mathbf{x}_n)$ of the data, where N is the number of data points. To solve this problem we follow the common approach to use the *expectation-maximization* (EM) algorithm [25]. The EM algorithm is initialized by the result of five iterations of K-means clustering. For both the initialization and the fitting of GMMs to data, we use the Netlab Matlab toolbox [24].

For novelty detection, we first use the GMM to compute the log probabilities $\log p(\mathbf{x}_i)$ of all training examples. To identify a test vector $\mathbf{f}$ as novel, its log probability $\log p(\mathbf{f})$ must be lower than the log probability of most training examples. Fig. 2 b) illustrates this method by comparing two histograms: the histogram of log probability of the training set and the histogram of log probability of test examples whose class is not in the training set. In practice, our terrain classification system computes a log probability threshold t_p below which test examples will be identified as novel. The higher t_p, the more samples belonging to unknown classes are be recognized as unknown, but also more (outliers) of known classes. The smaller t_p, the more test samples of unknown classes are misclassified as being known, but also more known samples are correctly accepted. Experimentally we found that good results can be obtained by setting t_p such that the log probabilities of 99% of the training data are above the threshold.

The K-means clustering, which initializes the EM algorithm, is initialized randomly. Depending on the random starting values, the minima found by the EM algorithm may be suboptimal, which can lead to suboptimal GMMs. Therefore, we use ten GMMs for novelty detection instead of a single one. Each of these GMMs gives an individual novelty prediction for a test example $\mathbf{f}$. A voting scheme then marks $\mathbf{f}$ as novel if more than half of the GMMs rejected $\mathbf{f}$.

2.2. Classification using kNN

As our system must be able to frequently update its classifier online, we use the k-nearest-neighbor algorithm, because its training is extremely fast. Additionally, despite its simplicity, we found that kNN performs very well for vibration-based terrain classification [26]. Training the kNN simply means to store all training vectors $\mathbf{x}_i$. To classify a test vector $\mathbf{f}$, the distances d_i of $\mathbf{f}$ to all $\mathbf{x}_i$ are computed and the k training vectors with the smallest distances are selected. The predicted class is the one that is most frequent among the selected vectors. If there is a draw, a random label is chosen from the winners. We use $k = 10$ in our experiments, because in [26], we found $k = 10$ to work best among $k \in \{1, 2, 5, 10, 15\}$. As source data, we use the PSD feature vectors of full dimension.

2.3. Update of a Known Class

Having classified a terrain segment $\mathbf{f}$ as a known class k_i, the system decides whether to update the training set with $\mathbf{f}$. Currently, our system has a fixed upper limit of 100 training examples per class. When a new class is created, however, this class typically has fewer members. If the robot traverses further examples of the new class, they are added to the training set until the new class reaches the maximum number of members. In order not to update the model by wrongly classified test examples, we only add test vectors whose predecessor was classified as the same class. If one changes the condition so that more than only the preceding terrain segment must be of the same class, less samples are added. These samples, however, are also more likely not to be outliers. Each time we update a known class, we must recompute the GMMs on the updated training set. Thus to save computation time, we do not update the model after each selected test sample, but whenever 10 such samples have been stored in a buffer.

2.4. Creation of a New Class

Before the system adds a new terrain class, a sufficiently large number q of examples of the unknown class must be available. Thus, a rejected terrain segment $\mathbf{f}$ is first stored in a buffer. To prevent outliers of known classes from being stored, $\mathbf{f}$ is only added if either the preceding or the successive example is also added. For the same reason, the system additionally possesses a forget mechanism. If no terrain segment is added to the buffer during a fixed time t (currently, we use $t = 10$ s), the oldest entry of the buffer is deleted.

As soon as the number of buffer entries exceeds q, a new terrain class is created from the buffer entries. In our experiments, we use $q = 20$, which enables fast learning of new classes. If there is a frequent change of terrain classes, this relatively small q also reduces the chance of creating new classes containing different terrain types. To update the model with the new class, three steps are necessary. First, the kNN must be retrained by simply adding the samples of the new class to the training set. Second, the PCA must be recomputed on the updated training set. Finally, new GMMs for novelty detection are created on the version of the updated training set that has been processed by PCA.

2.5. Estimation of Class Properties

When a new class is created, its physical properties are unknown, because there is no label like “asphalt” or “grass” indicating the properties. Therefore, our system tries to assign the properties “hardness” and “bumpiness” to the new class based on its raw vibration data. This section only shortly describes this procedure, because this is not the focus of this paper and our research in this direction is still in the beginning.

In experiments on raw vibration signals of different terrain surfaces, we identified some values computed from the signals that can serve as a hint for hardness and bumpiness. For example, the absolute value of the maximum and minimum of the y- and z-vibration typically is smaller the softer the surface, because soft surfaces dampen the vibration. An example of an indicating feature for the bumpiness is the number of sign changes in the x- or y-vibration signal. Lower numbers indicate bumpier surfaces. During big bumps, the acceleration stays positive or negative for a longer period, whereas for flat surfaces, the acceleration changes more rapidly between being positive and negative. These values are, however, often not characteristic enough when looking at a single terrain segment but only when averaged over a sufficiently large number of examples of a class.

Our current system computes a number of such values that measure hardness and bumpiness for each class. By comparison to a threshold, each measure casts a vote for hard or soft (or bumpy or flat). The largest number of votes per property finally leads to a label for the class, for example “hard and bumpy”.

3. Experimental Results

We acquired experimental data with our RWI ATRV-JR outdoor robot (Fig. 3 a). The robot is equipped with an Xsens MTi three-axis accelerometer working at 100 Hz. At speeds of around 0.5 m/s, the robot traversed four different ground surfaces: asphalt, gravel, grass and the surface of a boule court (Fig. 3 b). The number of terrain segments

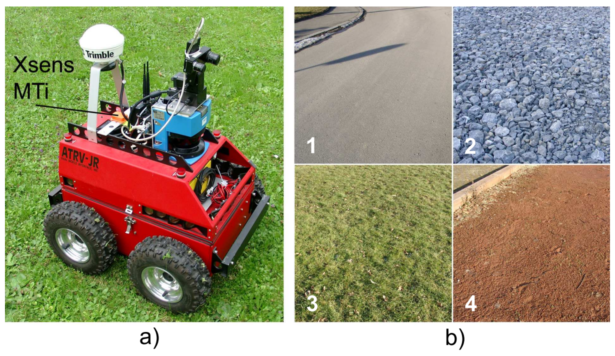
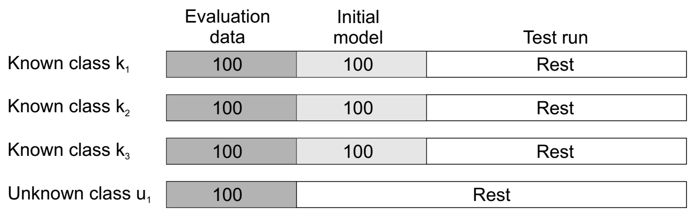


Figure 3. a) Our RWI ATRV-JR robot, equipped with an Xsens MTi sensor. b) Terrain types we used in the experiments: asphalt (1), gravel (2), grass (3), boule court (4).



	Evaluation data	Initial model	Test run
Known class k_1	100	100	Rest
Known class k_2	100	100	Rest
Known class k_3	100	100	Rest
Unknown class u_1	100	Rest	

Figure 4. Number of samples per class in the evaluation dataset, the initial model and the test run for the experiment with one unknown class.

is 513 for asphalt, 323 for gravel, 572 for grass and 579 for the boule court, which corresponds to about 33 minutes of robot drive. As depicted in Fig. 2 a), grass and gravel are relatively well seperated from the other classes. However, a considerable overlap exists between asphalt and boule court. This overlap does decrease, but it does not vanish, when more than two dimensions are used.

3.1. Experiments with One Unknown Class

In the first experiments we present in this paper, we trained the system on three classes and left one class unknown. Before starting an experiment, we split the data into parts (Fig. 4). Firstly, we put aside 100 evaluation examples per class. We classified the evaluation data at the end of the experiment using the final, automatically updated model, and compared the results to the predictions of a manually trained classifier which knows all classes from the start. Secondly, we use 100 examples of each known class to train the initial model. With the remaining data, we simulated a test run of the robot. In such a run, the system is first presented half the examples of each known class. Then, the system encounters half of the unknown examples. After that, the robot traverses the remaining half of all classes. In the first block of the test run, i.e. from the beginning until a new class is created from the unknown examples, most known examples should be accepted, and most unknown examples should be rejected. In the subsequent second block, all classes are known, so only a small number of test examples should be rejected. We executed experiments for all possible combinations of known and unknown classes. Additionally, we repeated each experiment five times with a permuted data order such that the training, test and evaluation sets are different each time.

Table 1. Average rejection rates (%) of the experiment with one unknown class ± standard deviation.

	Block 1	**Block 2**
Known examples	2.74 ± 0.34	3.51 ± 0.89
Unknown examples	71.01 ± 23.22	(no unknown class exists)

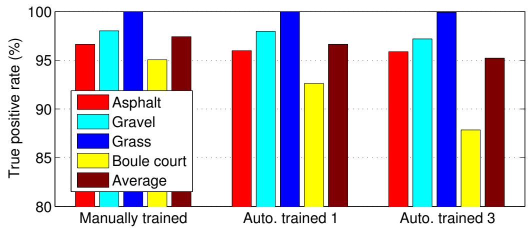


Figure 5. True positive rates on the evaluation dataset: manually trained (left) and automatically trained classifiers started with one (middle) or three (right) unknown classes.

Tab. 1 summarizes the percentage of test examples per block that were rejected by the novelty detector. From the known examples, only a small fraction was rejected. The rejection rate in the second block is a little larger than in the first block. This is because newly created classes consist of only 20 examples in the beginning. These 20 examples are not enough to explain almost all of the new classes's examples. For the examples belonging to unknown classes, the average rejection rate is over 70%. Without novelty detection, all of these examples would be accepted and classified wrongly. However, the large standard deviation of this value shows that the rejection rate differs greatly depending on which class is unknown. If there is little overlap with the known classes, the rejection rate is high, e.g. over 99% for grass. If there is a considerable overlap with one or more known classes, the rejection rate is lower, e.g. about 45% for asphalt.

Fig. 5 compares the mean true positive rate (TPR) on the evaluation dataset, achieved by the automatically updated system (96.65%), to the TPR of a manually trained kNN classifier (97.43%). These TPRs differ only slightly, as well as the rates for the individual classes. Only for the boule court, which has a considerable overlap with asphalt, there is a gap of about 2.5%. Note that in the automatically generated training set, newly added classes will only have little overlap with the other classes. The reason why the TPR on asphalt is better than on boule court is that some boule court test examples are spread in the main asphalt area, whereas much fewer asphalt test examples are in the main boule court region (Fig. 2 a). Note that all classes in our evaluation dataset have the same number of test samples. Therefore, the mean true positive rates correspond to the accuracy measure.

3.2. Experiments with Three Unknown Classes

In the second experiment, the system started with a single known class k_1 and three unknown classes u_1, u_2, u_3. We used the same data split and setup as in the first experiment, but with the following order of appearance of classes in the test run: $(k_1, u_1, k_1, \hat{u}_1, u_2, k_1, \hat{u}_1, \hat{u}_2, u_3, k_1, \hat{u}_1, \hat{u}_2, \hat{u}_3)$. $\hat{u}_i$ denotes that the i-th unknown class already has been added to the model, which should be the case if the system works well. We repeated the experiment for all 24 possible combinations of known and unknown classes, and repeated this five times for each combination with a permuted data order.

Table 2. Rejection rates (%) of the experiment with three unknown classes ± standard deviation.

	Block 1	**Block 2**	**Block 3**	**Block 4**
Known examples	6.83 ± 2.62	9.06 ± 2.11	7.73 ± 1.85	6.61 ± 1.21
Unknown examples	92.07 ± 10.56	85.65 ± 13.14	77.80 ± 15.83	-

Compared to the first experiment, the rejection rates for known classes are larger now (Tab. 2). The main reason is that during the experiment, the average number of training examples is smaller, because the three new classes start with only 20 examples. Thus, the explanatory power of the training set is lower on average. On the other hand, the rejection rates for unknown examples are also larger than in the first experiment. Again, the rejection rates vary depending on overlaps of the unknown class with known classes.

The mean TPR on the evaluation dataset using the final, automatically trained model is 95.22%, which is about 2% less than the TPR using a manually trained kNN (Fig. 5). Again, the boule court is classified significantly worse than the other classes.

On a 3 GHz Pentium 4 PC with 1 GHz RAM, we measured the following computation times for our Matlab code. When using the maximum number of training examples (400), computing the PCA took about 0.085 s. On average, novelty detection for one vector only required about 0.34 ms. kNN classification took about 4.9 ms per test vector. The time for training the kNN is negligible. Creating the novelty detector is the computationally most expensive part, because 10 Gaussian mixture models must be fitted to the data using the EM algorithm. When using maximally 20 iterations of the EM algorithm, fitting one GMM took about 0.160 s if the training set contains 400 examples. In this case, creating a new class (computing the PCA + creating the novelty detector) would take about 1.685 s. Using a C++ implementation, this should be possible in less time.

At the end of all experiments, the system updated all unknown classes to 100 examples. It assigned the following properties to the classes: asphalt is hard and flat, gravel is hard and bumpy, grass is soft and bumpy, and the boule court is hard and flat. Depending on the weather, the boule court could also be soft. As we recorded our data on a dry summer day, however, the boule court actually was hard and the assignment is correct.

4. Conclusion

We presented a vibration-based terrain classification system that is capable of identifying terrain classes that are not included in the training set. When the robot has collected a sufficiently large number of examples of the unknown terrain class, it adds a new terrain class online. Our experiments showed that a classifier trained by our system in an online fashion has a classification performance which is only slightly worse compared to a manually trained classifier to which all classes were known beforehand.

A problem for the current system appears if different unknown terrain types follow each other closely. Then, a new class will be possibly created containing more than one physical terrain type. We are currently searching for methods to detect such situations. Additionally, we will further improve our methods to assign properties to new classes. We will also investigate alternative novelty detection methods to GMMs. Furthermore, we want to implement the system in C++ instead of Matlab to achieve accelerated computation times for the model update.

References

[1] B. H. Wilcox, Non-Geometric Hazard Detection for a Mars Microrover, in *Proc. AIAA Conf. Intell. Robot. Field, Factory, Service, Space*, Houston, TX, USA, 1994, 675–684

[2] N. Vandapel, D. Huber, A. Kapuria and M. Hebert, Natural Terrain Classification using 3-D Ladar Data, in *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, New Orleans, LA, USA, 2004

[3] R. Manduchi, A. Castano, A. Talukder and L. Matthies, Obstacle Detection and Terrain Classification for Autonomous Off-Road Navigation, *Autonomous Robots* **18**, 2005, 81–102

[4] P. Bellutta, R. Manduchi, L. Matthies, K. Owens and A. Rankin, Terrain Perception for Demo III, in *Proc. IEEE Intelligent Vehicles Symposium*, Dearborn, MI, USA, 2000, 326–332

[5] A. Angelova, L. Matthies, D.M. Helmick and P. Perona, Fast Terrain Classification Using Variable-Length Representation for Autonomous Navigation, in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Minneapolis, USA, 2007, 1–8

[6] K. Iagnemma and A. Dubowsky, Terrain Estimation for High-Speed Rough-Terrain Autonomous Vehicle Navigation, in *Proc. SPIE Conf. on Unmanned Ground Vehicle Technology IV*, 2002

[7] E.M. DuPont, R.G. Roberts and C. Moore, The Identification of Terrains for Mobile Robots Using Eigenspace and Neural Network Methods, in *Proc. Florida Conf. on Recent Advances in Robotics*, Miami, Florida, USA, 2006

[8] C.A. Brooks and K. Iagnemma, Vibration-Based Terrain Classification for Planetary Exploration Rovers, *IEEE Transactions on Robotics* **21**(6), 2005, 1185–1191

[9] C. Weiss, H. Fröhlich and A. Zell, Vibration-based Terrain Classification Using Support Vector Machines, in *Proc. Int. Conf. on Intelligent Robots and Systems (IROS)*, Beijing, China, 2006, 4429–4434

[10] C. Weiss, M. Stark, A. Zell, SVMs for Vibration-based Terrain Classification, in *Proc. Autonome Mobile Systeme (AMS)*, Kaiserslautern, Germany, 2007, 1–7

[11] D. Stavens, G. Hoffmann and S. Thrun, Online Speed Adaptation Using Supervised Learning for High-Speed, Off-Road Autonomous Driving, in *Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI)*, Hyderabad, India, 2007

[12] R.E. Karlsen and G. Witus, Terrain Understanding for Robot Navigation, in *Proc. Int. Conf. on Intelligent Robots and Systems (IROS)*, San Diego, CA, USA, 2007, 895–900

[13] C.A. Brooks and K. Iagnemma, Self-supervised Terrain Classification for Planetary Rovers, in *Proc. NASA Science Technology Conference*, 2007

[14] I. Halacti, C.A. Brooks and K. Iagnemma, Terrain Classification and Classifier Fusion for Planetary Exploration Rovers, in *Proc. IEEE Aerospace Conference*, Big Sky, MT, USA, 2007

[15] D.M.J. Tax and R.P.W. Duin, Data Description in Subspaces, in *Proc. Int. Conf. on Pattern Recognition (ICPR)*, Barcelona, Spain, 2000, 672–675

[16] D. Dasgupta and F. Nino, A Comparison of Negative and Positive Selection Algorithms in Novel Pattern Detection, in *Proc. Int. Conf. on Systems, Man, and Cybernetics (SMC)*, Nashville, USA, 2000, 125–130

[17] G.C. Vasconcelos, M.C. Fairhurst and D.L. Bisset, Investigating Feedforward Neural Networks with Respect to the Rejection of Spurious Patterns, *Pattern Recognition Letters* **16**(2), 1995, 207–212

[18] B. Schölkopf, R. Williamson, A. Smola, J. Shawe-Taylor and J. Platt, Support Vector Method for Novelty Detection, in *Proc. Advances in Neural Information Processing Systems*, 2000, 582–588

[19] S. Marsland, U. Nehmzow and J. Shapiro, Environment-Specific Novelty Detection, in *Proc. Int. Conf. on Simulation of Adaptive Behaviour (SAB)*, Edinburgh, UK, 2002, 36–45

[20] M. Markou and S. Singh, Novelty Detection: a Review - Part 1: Statistical Approaches, *Signal Processing* **83**(12), 2003, 2481–2497

[21] M. Markou and S. Singh, Novelty Detection: a Review - Part 2: Neural Network Based Approaches, *Signal Processing* **83**(12), 2003, 2499–2521

[22] L. Tarassenko, P. Hayton, N. Cerneaz and M. Brady, Novelty Detection for the Identification of Masses in Mammograms, in *Proc. Int. Conf. on Artificial Neural Networks (ICANN)*, Paris, France, 1995, 442–447

[23] S.J. Hickinbotham and J. Austin, Neural networks for novelty detection in airframe strain data, in *Proc. IEEE Int. Joint Conf. on Neural Networks (IJCNN)*, Como, Italy, 2000, 375–380

[24] I.T. Nabney, *NETLAB: Algorithms for Pattern Recognition*, Springer, 2002

[25] A.P. Dempster, N.M. Laird and D.B. Rubin, Maximum Likelihood From Incomplete Data via the EM Algorithm, *Journal of the Royal Statistical Society* **39**(1), 1977, 1–38

[26] C. Weiss, N. Fechner, M. Stark and A. Zell, Comparison of Different Approaches to Vibration-based Terrain Classification, in *Proc. European Conf. on Mobile Robots (ECMR)*, Freiburg, Germany, 2007