Comparison of Different Approaches to Vibration-based Terrain Classification

Christian Weiss* Nikolas Fechner* Matthias Stark* Andreas Zell*
*Department of Computer Science, University of Tübingen, Tübingen, Germany

***Abstract*— There is a variety of different terrain types in outdoor environments, each posing different dangers to the robot and demanding a different driving style. In a previous paper, we presented a terrain classification method based on Support Vector Machines (SVM), which uses vibrations induced in the body of the robot to learn different terrain classes. However, in the previous paper, our experimental results were based on vibration data collected by a hand-pulled cart with relatively hard wheels. In this paper, we present experiments on data collected by our RWI ATRV-Jr outdoor robot. Additionally, we compare our SVM-based method to alternative classification methods. The comparison shows that our approach outperforms the other methods.**

***Index Terms*— Outdoor robotics, vibration-based terrain classification**

I. Introduction

In outdoor environments, a mobile robot typically faces many different terrain types. Some of them are flat and not slippery, and therefore the robot can traverse them at relatively high speed. Other ground surfaces are loose, slippery or bumpy, and therefore dangerous. To prevent accidents, the robot has to traverse these regions slowly and carefully. These examples show that the ground surface itself is a possible hazard to the robot in outdoor environments. Such a hazard is called a *non-geometric hazard* [21]. The robot can only avoid accidents if it adapts its driving style to the current terrain type.

One way to determine the terrain type is to directly estimate terrain parameters like cohesion or slippage from sensor measurements. Another way is to group the terrain into classes like asphalt, dirt or gravel, and to learn these classes from training examples. Once the robot has learned the different classes, it can classify new terrain data according to the learned model.

The most common data used for terrain classification are data collected by laser scanners or cameras. Ladar-based methods often focus on segmenting the ground surface from vegetation or all kinds of obstacles (e.g. rocks) instead of estimating the type of the ground surface itself [18, 8, 19, 12]. Other methods divide the ground surface in navigable and non-navigable regions [22]. Vision-based methods usually use texture or color information [1, 4, 12]. Some research has also been done on using force-torque sensors and potentiometers to detect non-geometric hazards [9, 11].

Vibration-based terrain classification was first suggested by Iagnemma and Dubowsky [10]. The idea is to measure the vibration that is induced in the robot while it traverses the terrain. The vibration can be measured at the wheels, the axes or the body of the robot. Usually, accelerometers are used to measure the vibration perpendicular to the ground surface (z-acceleration). As different terrain types induce different vibration signals, one tries to learn characteristic vibration signals for each terrain type from training examples. The learned model is then used for classification of unknown data. The disadvantage of the method is that terrain can be classified only while the robot traverses it, but not beforehand. Advantages are, for example, the independence from illumination conditions and the high reliability. Thus, vibration-based terrain classification can be used as a stand-alone classifier or in combination with other sensors.

Brooks and Iagnemma examined vibration-based terrain classification for planetary rovers [2, 3]. They use Principal Component Analysis (PCA) to reduce the dimensionality of their data and Linear Discriminant Analysis (LDA) for classification. Sadhukhan and Moore presented an approach based on probabilistic neural networks (PNN) [14, 15]. In [20], we suggested an approach that uses Support Vector Machines (SVM) for classification. Stavens et al. presented an approach for vehicles driving up to 35 mph [17]. However, they focused on assessing the roughness of the terrain to adapt the velocity, and not on grouping the ground surface into classes.

In our previous paper [20], we obtained our experimental results using data from a hand-pulled cart with relatively hard wheels. These wheels lead to relatively clear and strong vibration signals. In this paper, we present experimental results on data that we collected using our RWI ATRV-Jr outdoor robot. Its big, air-filled tires are likely to dampen the vibration signals. We also examine how different robot speeds influence the classification performance. Additionally, we implemented the terrain classification approach presented by Sadhukhan and Moore as well as the approach suggested by Brooks and Iagnemma, and compare both to our approach. These methods cover two of the four main groups of classification methods, namely kernel methods (our SVM-based approach) and neural networks (the PNN). From the third group, the methods based on Likelihood, we chose Naïve Bayes, which is a standard method from this group. The fourth group are decision trees, from which we examined the J4.8 algorithm. J4.8 is based on the well known C4.5 algorithm. Finally, we also tested a k-nearest-neighbor (kNN) classifier.

The rest of this paper is organized as follows. Section II recapitulates our SVM-based terrain classification approach. Section III briefly describes the alternative classification methods. Section IV presents our experimental results and finally, Section V concludes the paper and suggests future work.

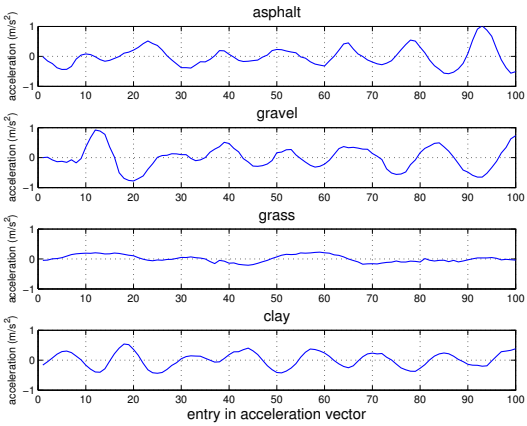

Fig. 1. Some example acceleration vectors for different terrain types.

II. SVM-based Terrain Classification Method

In this section, we first give an overview over our method. Then we present feature extraction and SVM classification in more detail.

A. Method Overview

Our method consists of a training phase and a classification phase. In the training phase, we first drive the robot around on known terrain to collect z-acceleration data at 100 Hz. Then, we split the acceleration into 1×100 vectors, i.e. each vector represents 1 s of robot travel. Additionally, we label each vector with its terrain type. Then we transform each vector to the frequency domain (Section II-B). Next, we normalize each feature (= frequency component) to mean 0 and standard deviation 1. In the last step, we train an SVM on the labeled feature vectors (Section II-C). The training phase is an offline step, because it is computationally intensive.

In the online test phase, the robot drives around on unknown terrain. After each second, we create a 1×100 acceleration vector, transform it to the frequency domain and normalize it using the same parameters used during training. Then, we classify the resulting test vector using the trained SVM to get the predicted terrain type.

B. Feature Extraction

The vibration vectors collected by the robot contain acceleration values measured perpendicular to the ground surface. Fig. 1 shows some examples. Except for grass, the signals are very similar. Thus, it is beneficial to transform these vectors to a more significant representation.

In [20], we compared different representations: a Fast Fourier Transform (FFT) representation as suggested by Sadhukhan [14], a log-scaled power spectral density (PSD) as used by Brooks and Iagnemma [3], and a more compact representation based on simple features calculated from the acceleration vector (e.g. the number of sign changes). In our experiments using data of the hand-pulled cart, we found the simple features to work best. However, for the data collected by our robot, the frequency-based representations, i.e. the FFT and the log-scaled PSD, work better. Thus, for the rest of this paper, we will use either a log-scaled PSD or a 128-point FFT of the acceleration data as feature vector.

C. SVM Classification

After feature extraction, we use an SVM [6] to learn for each terrain type a separation from all other terrain types (*one-versus-rest classificaton*). Later on, an unseen test pattern will be assigned to that class, for which the distance to the decision boundary is largest.

SVMs belong to the family of kernel methods [16]. The idea is to construct a separating hyperplane between two classes of points, such that the margin between the hyperplane and the points closest to it becomes maximal. Nonlinear classification can be achieved by first mapping the original data to some high dimensional feature space in a nonlinear fashion. This computation is usually done implicitly by means of a kernel function, which defines a dot product between points in feature space. It is also possible to allow for a small number of training errors by means of a so-called soft margin parameter C that regularizes the trade-off between maximizing the margin and minimizing the training error.

In our case we employ a Radial Basis Function (RBF) $k(x,y) = \exp(-\|x-y\|^2/2\sigma^2)$ as kernel function, where x and y are two feature vectors. The width σ of the RBF kernel together with the soft margin parameter C are tuned via a systematic search on the grid $\log_2\sigma \in \{\hat{\sigma}/4, ..., 4\hat{\sigma}\}$ and $\log_2 C \in \{-2, ..., 14\}$, where $\hat{\sigma}$ is set such that $\exp(-D/2\hat{\sigma}^2) = 0.1$. D denotes the length of the feature vectors. Each candidate parameter vector (σ, C) on the grid is evaluated by 5-fold cross-validation. Note that this automatic parameter selection only involves the training data, but not the test data. As SVM implementation we use LIBSVM [5].

III. Alternative Classification Methods

This section briefly describes the classification methods to which we compare our SVM-based approach. The FFT and log-scaled PSD feature vectors are the same for all approaches.

A. PNN

Using probabilistic neural networks for terrain classification was suggested by Sadhukhan [14]. As feature vector, he used the frequency components of a 1024-point FFT that are between 10 and 20 Hz. However, in our experiments, we used a 128-point FFT and all frequency components, because this led to better results. Please note that we did not compute the results presented in Section IV using Sadhukhan's original code given in [14], but using an own implementation of PNNs. The reason is that on average, our code produced better results than Sadhukhan's code.

A PNN has three layers (Fig. 2). The first layer is the input layer and consists of d *input units* a_i, where d is the dimensionality of the data. Each input unit is connected to each of the n *pattern units* $\mathbf{w_i}$, where n is the number of training vectors. Each of the pattern units is connected to a single *category unit* c_i. The number c of category units corresponds to the number of classes.

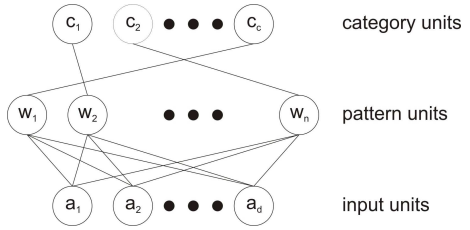

Fig. 2. A probabilistic neural network consisting of d input units a_i, n pattern units $\mathbf{w_i}$ and c category units c_i.

In the training phase, the n normalized training vectors $\mathbf{x_i}$ are presented to the PNN. Basically, training consists of storing each training vector $\mathbf{x_i}$ in the corresponding pattern unit $\mathbf{w_i}$. Additionally, each pattern unit is connected to the category unit corresponding to the label of the training vector.

For classification, a normalized test vector $\mathbf{y}$ is presented to the PNN. Each pattern unit $\mathbf{w_i}$ calculates the inner product $net_i = \mathbf{w_i}^t\mathbf{y}$ which is called the *net activation*. Then, the pattern unit $\mathbf{w_i}$ emits the value $e^{(net_i-1)/\sigma^2}$ of a nonlinear *activation function* to the attached category unit c_k. The parameter σ must be set by the user. In our experiments, we found $\sigma = 0.4$ to work best. Each category unit sums the inputs from the pattern units. Finally, the category unit with the highest accumulated value gives the predicted label. A more detailed description of PNNs can be found in [7].

B. Brooks's Method

In the approach presented by Brooks and Iagnemma, the terrain can also be labeled as "unknown", if the classifier is not sure about the predicted class. We implemented the algorithm based on the detailed descriptions given in [2] and [3].

Brooks and Iagnemma transform their acceleration data to a power spectral density (PSD). A log-scaling of the magnitude reduces the dominating effect of high-magnitude frequency components. Then, they use principal component analysis to reduce the dimensionality of their feature vectors and to separate the signal from noise. They suggest to use $k = 15$ principal components. However, we used $k = 30$, because this worked better for our data.

To separate feature vectors of different classes, Brooks and Iagnemma use linear discriminant analysis (LDA). They train a set of pairwise classifiers, one classifier for each possible pair of terrain types. These classifiers take into account both the distribution of feature vectors within a single class as well as the separation of the class means, and compute a discrimination vector $\mathbf{d}$. To classify a test vector $\mathbf{y}$, the dot product $d(\mathbf{y}) = \mathbf{dy}$ is computed and the Mahalanobis distance of $d(\mathbf{y})$ to both terrain class means is calculated. If the difference between the Malahanobis distances is < 1, the terrain class of $\mathbf{y}$ is "unknown" and both involved classes get an "unknown" vote. Otherwise, the class mean with the smaller distance specifies the predicted label and casts a positive vote for the winning class and a negative vote for the other class.

If there are more than two classes, a voting scheme sums up the votes of the pairwise classifiers. The class with the largest number of positive votes wins. If the number of "unknown" votes for this class is equal to or larger than the number of positive votes, or there are one or more negative votes, the terrain is classified as "unknown".

C. kNN

The k-nearest-neighbor algorithm is a very simple classification method. However, it often performs very well and therefore, it is an important benchmark method.

For training, simply all training vectors $\mathbf{x_i}$ are stored. To classify a test vector $\mathbf{y}$, the distances d_i of $\mathbf{y}$ to all training vectors 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 labels of the selected vectors. If there is a draw, often simply a random label is chosen from the winners. In our experiments, we found $k = 10$ to work best among the values $k \in \{1, 2, 5, 10, 15\}$.

D. Decision Trees

Decision Trees are classical machine learning approaches based on information theoretical considerations. The version J4.8 used in this paper is a Java implementation based on the well known C4.5 algorithm [13]. Both of them rely on the ID3 algorithm and use entropy-based measures for constructing the classification tree. The *entropy* $E(X) = \sum_i -p_i \log_2 p_i$ is a measure for the disorder of a data collection X with p_i being the proportion of X belonging to the class i. This measure can now be used for defining the ability of an attribute A for classifying the examples by calculating the expected reduction of entropy if the data are ordered according to A. This leads to the *information gain* $G(X, A) = E(X) - \sum_{v \in Values(A)} \frac{|X_v|}{|X|} E(X_v)$ which describes the discriminatory power of an attribute A.

The ID3 algorithm constructs the tree top-down by choosing the most descriptive attribute A in each node H and adding a new branch to H for each value of A in the case that the attribute is nominal. Otherwise a binary split according to the value of A is performed. Finally, each leaf is labeled by the most common class of the subset of X represented by it.

E. Naïve Bayes

In contrast to the previously described approaches, Naïve Bayes [13] does not perform a simple prediction but a probability estimation of a class. The basic idea of this algorithm is to assign each instance $\mathbf{x}$ the class v_i which is the most probable class considering its attributes A_i. Using Bayes' theorem, this can be expressed as $c(\mathbf{x}) = argmax_{v_i}\ P(a_1, ..., a_n|v_i)P(v_i) = argmax_{v_i}\ P(v_i)\prod_i P(a_i|v_i)$. The training of the classifier simply consists of estimating the $P(v_i)$ and $P(a_i|v_i)$ based on the distribution in the training data.

IV. Experimental Results

To collect vibration data, we used our RWI ATRV-Jr outdoor robot (Fig. 3). We mounted an Xsens MTi sensor on an aluminium plate on top of the robot. The Xsens MTi measures the z-acceleration perpendicular to the ground floor at 100 Hz.

Fig. 3. Our RWI ATRV-JR outdoor robot "Arthur".

Fig. 4. The different terrain types we used in our experiments: 1) indoor floor 2) asphalt 3) gravel 4) grass 5) paving 6) clay.

TABLE I
NUMBER OF SAMPLES PER CLASS IN OUR DATASET

class	0.2 m/s	0.4 m/s	0.6 m/s	total
indoor floor	282	549	581	1412
asphalt	499	513	600	1612
gravel	311	323	392	1026
grass	482	572	631	1685
paving	314	573	567	1454
clay	423	579	605	1607
no motion	199	615	615	1429
total	2510	3724	3991	10225

In the middle of July and in the beginning of December, we collected vibration data by driving the robot over six different terrain types: indoor PVC floor, asphalt, gravel, grass (i.e. the soil under the grass), paving and clay (the surface of a boule court). As seventh "terrain type", we added some data from situations in which the robot did not move. Fig. 4 shows example images of the different surfaces. Additionally, we used three different robot speeds: about 0.2 m/s, 0.4 m/s and 0.6 m/s. In total, our dataset contains 10225 samples which correspond to about 2 h 50 min of robot drive. Tab. I shows the number of samples in our dataset in more detail.

In our experiments, we used 10-fold cross-validation. This means that for each experiment, we split the data into 10 equally sized parts. In each of the 10 folds, we used 9 parts for training, and the remaining part for testing. We obtained the final result by averaging the results of the folds.

As quality measures of a classification result, we use the *true positive rate* (TPR) and the *false positive rate* (FPR). The TPR for a class c_i is the percentage of test vectors belonging to c_i that were correctly classified as class c_i. The FPR for class c_i is the percentage of test vectors not belonging to c_i but wrongly classified as class c_i. Good results are characterized by a high TPR and a low FPR.

It is difficult to compare the classification method of Brooks to the other approaches, because it is the only one allowing "unknown" predictions. These "unknowns" are not counted as correct, so the TPR of Brooks's method is expected to be lower than for the other methods. However, the "unknowns" are also not regarded as false. Thus, the FPR is expected to be better than for the other methods.

For the different classifiers, often one of the FFT or log-scaled PSD representation works better than the other. Thus, for each classification method, we present only the results obtained by using the more suitable features. For PNN, Naïve Bayes and J4.8, we used the FFT representation, whereas for kNN and the method of Brooks, we used the log-scaled PSD. For the SVM, we present the results for both features.

In a first experiment, we considered only the three terrain classes gravel, grass and clay, because there may be many environments were only a small number of different terrains exist. Fig. 5 shows the true positive rates obtained by the different methods at different speeds. Additionally, the figure shows the results when all speeds are merged in one dataset. Note that the TPRs in Fig. 5 are the average TPRs of the three classes. The results of the SVM lie between 95 and 98% for 0.2 to 0.6 m/s and between about 91 and 92% for mixed velocities. Only the kNN method performs similarly well. The performance of the PNN strongly depends on the velocity; the TPRs are between 72 and 85%. The TPRs using Brooks's method, Naïve Bayes and J4.8 are around 90% for the individual velocities. However, on mixed velocities, the TPRs drop to about 77-82%. The results show no general trend on which velocity can be classified best. However, all methods perform better on individual velocities than on the mixed dataset.

Fig. 6 shows the false positive rates for the 3-class experiment. As expected, the method of Brooks outperforms all other approaches. This method prefers classifying samples as "unknown" to classifying them wrong. Among the other approaches, SVM and kNN perform similarly well. PNN, Naïve Bayes and J4.8 perform significantly worse.

In [3], Brooks and Iagnemma also presented results of a 3-class experiment including sand, dirt and gravel. They used vibration data collected at 44.1 kHz by the rover TORTOISE, whose velocity varied from 2 to 5 cm/s, and split the vibration data into segments of 3 s. About 85.3% of the test vectors were correctly classified in their experiment, and about 10.7% of the samples were classified as "unknown". Despite the very different experimental settings in our experiment, Brooks's method also performs well. It achieves a a TPR of over 90% on individual velocities, and a TPR of 77% on the mixed data.

In [14], Sadhukhan presented a 3-class experiment that is very similar to our experiment. An RWI ATRV-Jr robot collected acceleration signals at 100 Hz. The robot traversed a testbed consisting of grass, dirt and gravel at 0.2, 0.4, 0.6 and 0.8 m/s. Sadhukhan observed that the classification rates

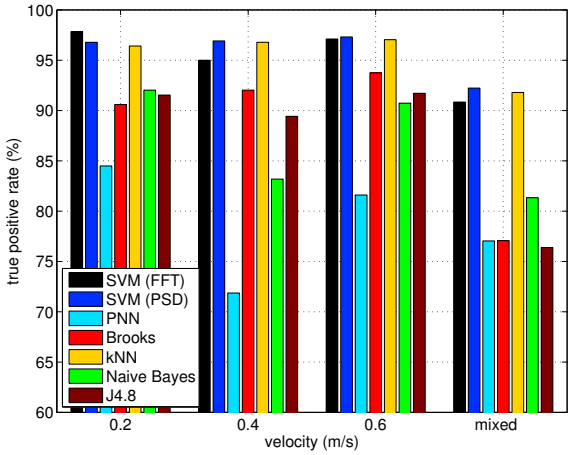

Fig. 5. True positive rates in the 3-class experiment.

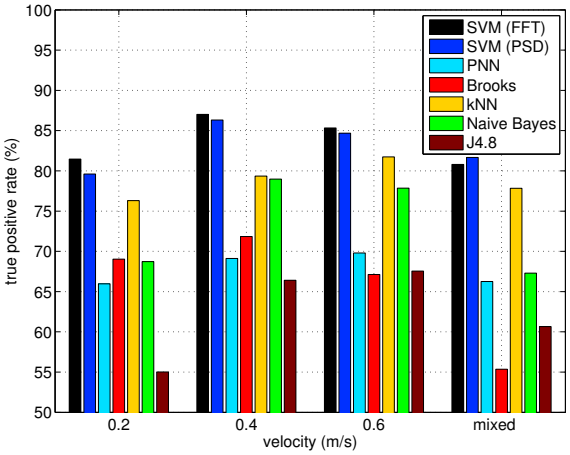

Fig. 7. True positive rates in the 7-class experiment.

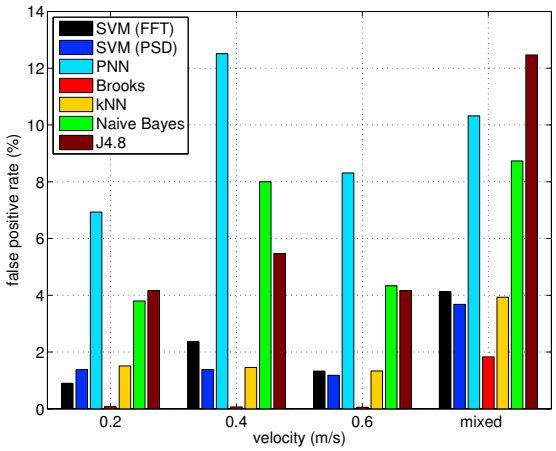

Fig. 6. False positive rates in the 3-class experiment.

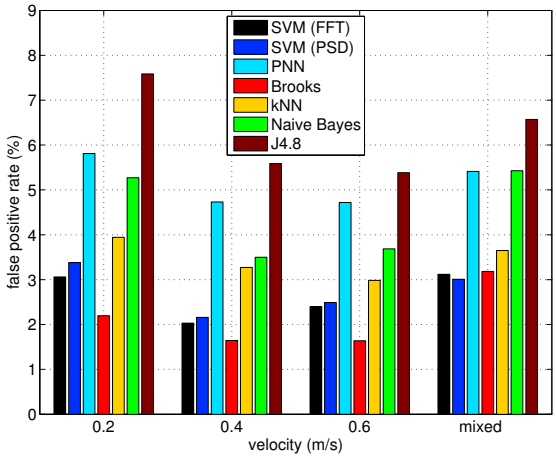

Fig. 8. False positive rates in the 7-class experiment.

increase with the speed of the robot. At 0.2 m/s, about 75% of the test vectors were classified correctly, whereas the TPR was about 94.7% at 0.8 m/s. Our experiments did not confirm this finding, perhaps because Sadhukhan collected his data in a more controlled testbed environment.

In a further experiment, we included all seven terrain classes. The classification performance of all approaches decreases due to the more difficult dataset. Fig. 7 shows that again our SVM approach performs best. The TPRs for 0.2 m/s and the mixed dataset are about 80%, and for 0.4 and 0.6 m/s, the TPRs are about 85-86%. This time, the kNN can not match the performance of the SVM, but is best among the other approaches. In total, Naïve Bayes is next best, followed by PNN, the method of Brooks and the J4.8 decision tree. Using mixed velocities, the approach of Brooks classifies about 26% of the samples as "unknown", which lead to a low TPR. The results also show that data collected at 0.2 m/s are more difficult to classify than data collected at higher speeds.

Fig. 8 shows the false positive rates of the 7-class experiment. The approach of Brooks again performs best. However, the difference to the SVM is smaller than in the 3-class experiment. On mixed velocities, the FPRs of Brooks's method and our SVM method are approximately equal. The further ranking is kNN, followed by Naïve Bayes, PNN and J4.8.

We also examined how well different terrain classes could be classified. For this purpose, Tab. II shows the confusion matrix of our SVM method. The entry (i, j) of the matrix shows how often (in %) samples belonging to class i were classified as class j. To create the confusion matrix, we used our most difficult dataset: seven terrain classes and mixed velocities.

The confusion matrix indicates that a stopped robot is correctly detected in most of the cases. Indoor floor is wrongly classified as asphalt in about 20% of the cases. In turn, asphalt is misclassified as indoor floor in about 12% of the cases. About 17.5% of gravel samples are wrongly classified as paving and nearly 10% of paving is classified as gravel. These pairs show that the most noticeable misclassifications occur between terrain types that are rather similar. Grass seems to be well distinguishable from all other types.

Tab. III compares the computation times of the different classifiers on two example datasets A and B, measured on a 3 GHz PC with 1 GB of RAM. In the 3-class dataset A, we did training on 3886 vectors and classification on 432 vectors. In the 7-class dataset B, we used 9203 vectors for training and 1022 vectors for classification. Both datasets contain mixed

TABLE II

CONFUSION MATRIX FOR 7-CLASS CLASSIFICATION USING SVMS. NM: NO MOTION, IN: INDOOR FLOOR, AS: ASPHALT, GV: GRAVEL, GS: GRASS, PV: PAVING, CL: CLAY.

	NM	**IN**	**AS**	**GV**	**GS**	**PV**	**CL**
NM	**99.93**	0.07	0	0	0	0	0
IN	0	**71.66**	18.28	0.07	0.35	0.99	8.64
AS	0	11.85	**74.13**	0.43	0.50	3.91	9.18
GV	0	0.10	0.29	**70.37**	10.44	17.55	1.27
GS	0	0	0.06	3.02	**93.83**	1.96	1.13
PV	0	1.03	2.75	9.63	3.30	**76.55**	6.74
CL	0	2.55	7.41	0.87	2.93	7.15	**79.09**

TABLE III

COMPUTATION TIMES ON 3-CLASS DATASET A AND 7-CLASS DATASET B

	Training (s)		Classification (ms)	
	A	B	A	B
Brooks	14.104	100.670	0.047	0.150
Naïve Bayes	0.473	1.360	1.420	2.250
SVM (PSD)	3223.729	27213.897	0.835	3.086
SVM (FFT)	4251.225	29820.799	0.978	4.125
J4.8	8.990	41.750	19.221	38.612
kNN	$7{\cdot}10^{-6}$	$7{\cdot}10^{-6}$	27.076	65.218
PNN	0.054	0.130	66.464	135.692

velocities. Note that Tab. III shows the training times for complete training, whereas classification times are average values for a single vector.

The training times for the SVM (up to several hours) are much higher than for the other methods, because the gridsearch to tune the parameters is very time-consuming. As training can be done offline, however, the time needed to classify a test vector is more important. Using Naïve Bayes, the SVM and the method of Brooks, classification takes at most a few milliseconds. The classification times for J4.8, kNN and PNN are significantly higher.

V. CONCLUSION

In this paper, we presented extensive experiments on vibration-based terrain classification. The experiments showed that our SVM-based method works well on data collected by a common outdoor robot driving between 0.2 and 0.6 m/s. As the classification rates tended to slightly increase with higher speed, we think that our method will also work for velocities of about 1 m/s or above. However, the more different velocities are merged together in one dataset, the more difficult the classification. A solution could be to train separate classifiers for different ranges of velocities, and to select the appropriate classifier for a test vector based on the velocity measured by the robot's odometry.

We also compared our method to other approaches: an method suggested by Brooks and Iagnemma, an approach presented by Sadhukhan and Moore based on probabilistic neural networks (PNN), a k-nearest-neighbor classifier, a Naïve Bayes approach and a J4.8 decision tree. The experiments showed that among the compared approaches, our SVM-based method worked best. Despite its simplicity, the kNN classifier also worked well. Unlike the other methods, the approach of Brooks and Iagnemma can classify a sample as "unknown", which often led to a low true positive rate, but also to a low false positive rate. PNN, Naive Bayes and the J4.8 decision tree performed worse than the other approaches.

In the future, we plan to examine if it is possible to learn a separation between terrain classes from scratch without prior knowledge. Additionally, we will investigate if the acceleration measured in other directions, e.g. sideways, is also suitable to capture the vibration, and if a combination of different directions can further improve the classification rates.

REFERENCES

[1] P. Bellutta, L. Manduchi, K. Matthies, K. Owens, and A. Rankin. Terrain perception for demo III. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 326 – 332, Dearborn, MI, 2000.

[2] C. Brooks, K. Iagnemma, and S. Dubowsky. Vibration-based terrain analysis for mobile robots. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 3426–3431, Barcelona, Spain, 2005.

[3] C. A. Brooks and K. Iagnemma. Vibration-based terrain classification for planetary exploration rovers. *IEEE Transactions on Robotics*, 21(6):1185 – 1191, December 2005.

[4] R. Castano, R. Manduchi, and J. Fox. Classification experiments on real-world textures. In *Workshop on Empirical Evaluation in Computer Vision*, Kauai, HI, 2001.

[5] C. Chang and C. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/˜cjlin/libsvm.

[6] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273 – 297, 1995.

[7] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley, 2001.

[8] M. Hebert and N. Vandapel. Terrain classification techniques from ladar data for autonomous navigation. In *Proc. Collaborative Technology Alliances conference*, 2003.

[9] K. Iagnemma, C. Brooks, and S. Dubowsky. Visual, tactile, and vibration-based terrain analysis for planetary rovers. In *Proc. IEEE Aerospace Conference*, 2004.

[10] K. Iagnemma and S. Dubowsky. Terrain estimation for high-speed rough-terrain autonomous vehicle navigation. In *Proc. SPIE Conference on Unmanned Ground Vehicle Technology IV*, 2002.

[11] K. Iagnemma, S. Kang, H. Shibly, and S. Dubowsky. Online terrain parameter estimation for wheeled mobile robots with application to planetary rovers. *IEEE Transactions on Robotics*, 20(5):921 – 927, 2004.

[12] R. Manduchi, A. Castano, A. Talukder, and L. Matthies. Obstacle detection and terrain classification for autonomous off-road navigation. *Robotics and Automation*, 18:81 – 102, 2005.

[13] T. T. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[14] D. Sadhukhan. Autonomous ground vehicle terrain classification using internal sensors. Master's thesis, Dept. Mech. Eng., Florida State University, Tallahassee, Florida, USA, 2004.

[15] D. Sadhukhan and C. Moore. Online terrain estimation using internal sensors. In *Proc. Florida Conf. on Recent Advances in Robotics*, Boca Raton, FL, 2003.

[16] B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

[17] D. Stavens, G. Hoffmann, and S. Thrun. Online speed adaptation using supervised learning for high-speed, off-road autonomous driving. In *Proc. International Joint Conference on Artificial Intelligence (IJCAI)*, Hyderabad, India, 2007.

[18] A. Talukder, R. Manduchi, A. Rankin, and L. Matthies. Fast and reliable obstacle detection and segmentation for cross-country navigation. In *Proc. IEEE Intelligent Vehicles Symposium*, Versailles, France, 2002.

[19] N. Vandapel, D. Huber, A. Kapuria, and M. Hebert. Natural terrain classification using 3-d ladar data. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, New Orleans, LA, 2004.

[20] C. Weiss, H. Fröhlich, and A. Zell. Vibration-based terrain classification using support vector machines. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4429 – 4434, Beijing, China, October 2006.

[21] B. H. Wilcox. Non-geometric hazard detection for a mars microrover. In *Proc. AIAA Conf. Intell. Robot. Field, Factory, Service, Space*, volume 2, pages 675 – 684, Houston, TX, 1994.

[22] D.F. Wolf, G. S. Sukhatme, D. Fox, and W. Burgard. Autonomous terrain mapping and classification using hidden markov models. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, Barcelona, Spain, 2005.