Proceedings of Australasian Conference on Robotics and Automation, 2-4 Dec 2013, University of New South Wales, Sydney Australia

Terrain Classification Using a Hexapod Robot

Graeme Best*†, Peyman Moghadam*, Navinda Kottege*, Lindsay Kleeman†
*Autonomous Systems Laboratory, CSIRO Computational Informatics,
1 Technology Court, Pullenvale QLD, Australia
{graeme.best, peyman.moghadam, navinda.kottege}@csiro.au
†Department of Electrical and Computer Systems Engineering, Monash University,
Wellington Road, Clayton VIC, Australia
{graeme.best, lindsay.kleeman}@monash.edu.au

Abstract

The effectiveness of a legged robot's gait is highly dependent on the ground cover of the terrain the robot is traversing. It is therefore advantageous for a legged robot to adapt its behaviour to suit the environment. In order to achieve this, the robot must be able to detect and classify the type of ground cover it is traversing. We present a novel approach for ground cover classification that utilises position measurements of the leg servos to estimate the errors between commanded and actual positions of each joint. This approach gives direct insight into how the robot is interacting with the terrain. These position sensors are usually built into the actuators and therefore our approach has the advantage of not requiring any additional sensors. We employ a multi-class Support Vector Machine with a 660-dimensional feature space consisting of features in gait-phase and frequency domains. We implemented our algorithm in the Robot Operating System (ROS) framework for real time classification and also developed a MATLAB implementation for extensive offline testing. Both implementations perform multi-class ground cover classification with high accuracy across five classes.

1 Introduction

Humans naturally adjust their walking style to suit the terrain they are walking on. For example we intuitively know from our past experience to walk slowly and carefully on slippery ice but can run fast on grass. A legged robot could exhibit similar behaviour and then be effective at traversing a variety of terrains. In order for a robot to achieve this behaviour, it must have the ability to autonomously differentiate terrain types. Terrain classification is the process of identifying a patch of terrain as one of several predefined classes, such as grass, rocks, concrete and mulch. The amphibious legged robot presented by [Giguère *et al.*, 2006] exhibits a good example of this desired behaviour by differentiating between sand and shallow water to determine when to switch from a walking to swimming gait.

Terrain classification methods can be classified into two main categories: exteroceptive sensing and interactive sensing. Exteroceptive sensing based terrain classification algorithms use a variety of sensors such as vision [Vernaza *et al.*, 2008; Lu *et al.*, 2009; Moghadam and Wijesoma, 2009; Moghadam *et al.*, 2010; Filitchkin and Byl, 2012] and range [Saitoh and Kuroda, 2010; Belter and Skrzypczynski, 2011; Wang *et al.*, 2012b] to perceive and predict the type of terrains near the robot. However, these approaches have several shortcomings making them unsuitable for some systems. For example, poor results can be caused by large variations in lighting or visual characteristics of a terrain. Additionally, this type of sensing does not give insight into how the robot is currently interacting with its environment.

Interactive sensing techniques measure aspects of the interaction between the robot and the terrain as the robot moves through the environment. This gives the robot useful information regarding how its performance is currently being affected by the environment and therefore provides complementary information to exteroceptive sensing in many situations [Brooks and Iagnemma, 2012]. Vibration based interactive sensing modalities are the most common for terrain classification by wheeled robots [Sadhukhan, 2004; Ojeda *et al.*, 2006; Weiss *et al.*, 2006; DuPont *et al.*, 2008; Brooks and Iagnemma, 2012; Wang *et al.*, 2012a]. The body of legged robots experience similar vibrations or 'bouncing' and measurements of these movements can be used for terrain classification [Larson *et al.*, 2005; Giguère *et al.*, 2006].

However, for terrain classification by a legged robot, it is more intuitive to measure the effect on the legs rather than the body since the legs directly interact with the terrain. This can be done by adding force or torque sensors to the robot's legs or actuators. [Höpflinger *et*

al., 2010] presents this approach using a single vibrating robot leg detached from the body. Their experiment requires a scratching leg motion which is different to typical legged robot gaits and therefore would make it difficult to directly apply their technique. This issue is rectified in [Schmidt and Walas, 2013] who uses force and torque sensors attached to one leg of a hexapod robot while performing a standard gait. Both of these studies performed testing in a controlled laboratory environment and therefore they have not shown how well their algorithms generalise to realistic terrains where typically there are larger variations within each terrain class.

All of the terrain classification approaches in the previously mentioned studies require adding additional external sensors to the robot. These sensors can come at a cost of more power, input/output ports and computer processing requirements. We propose a new approach that utilises position sensors that come built into a hexapod robot's leg actuators and therefore does not require modifying the hardware of the robot. We use the position sensors to measure the gait control loop error for the joint angles, which provides similar information to the force and torque interactive sensing techniques [Höpflinger *et al.*, 2010; Schmidt and Walas, 2013]. To the best of our knowledge, this is the first study to perform terrain classification solely on servo position sensors using a multi-legged robot. To achieve this, we use a novel feature extraction method which exploits the periodic nature of a legged robot's gait by transforming the time signals into gait-phase domain. This provides unique insight into the interaction between the robot and the terrain. Our approach utilises sensors on the front two legs of the hexapod and therefore perceives a larger terrain patch to reduce the effect of any local variations of the terrain types. Additionally, unlike previous work [Höpflinger *et al.*, 2010; Schmidt and Walas, 2013], we use measurements from multiple joints on each leg and therefore have a more complete picture of each leg's interaction with the terrain. We show how our approach generalises to new terrains by performing training and testing with realistic outdoor terrains from multiple locations.

Figure 1: PhantomX Hexapod in Outdoor Environment

This paper is structured as follows. Section 2 describes the system used and section 3 details our proposed terrain classification algorithm. Section 4 discusses the results of testing we have performed using realistic natural terrains. Finally, section 5 concludes the paper followed by discussion of future directions.

2 Hexapod Robot System

We use a PhantomX AX Hexapod robot[1], shown in Figure 1, which features 18 Dynamixel AX-12 smart servomotors (3 for each leg, comprising coxa, femur and tibia servos). The position sensors built into the servos form the basis of our classification algorithm. The robot is equipped with an Arduino microprocessor board and accompanying software that includes an open-loop gait engine. For our tests, we use the standard tripod gait moving at approximately 25 cm/s with a 900 ms gait-cycle. All servo parameters are set to their default values. We have added to the software a mechanism for reading position sensor data from the six servos on the front two legs at 20 Hz. All sensor data is time-stamped when it is received by the Arduino board. The data is then collected in real time by a laptop connected via a serial connection, where the data can either be used for online classification with our Robot Operating System (ROS) implementation or saved into rosbag files for offline testing in either our ROS or MATLAB implementations.

3 Classification Algorithm

Figure 2 shows the stages of the classification algorithm, including collecting the raw data, training the SVM (left) and estimating the class of new observations (right), and these stages are described in detail in the following sections. The algorithm is essentially the same for our MATLAB offline implementation and ROS online implementation. However, in the ROS implementation the algorithm is split into several ROS nodes (processes) to allow parallel execution of the data collection and class prediction in real time whereas the MATLAB implementation runs sequentially.

3.1 Sensors

The algorithm uses the actual position and goal position feedback of the AX-12 smart servos. Figure 3 shows this feedback from the front left tibia servo (the servo furtherest from the body on the front left leg) over a 2.7 s

[1] `http://www.trossenrobotics.com/phantomx-ax-hexapod-mk1.aspx`

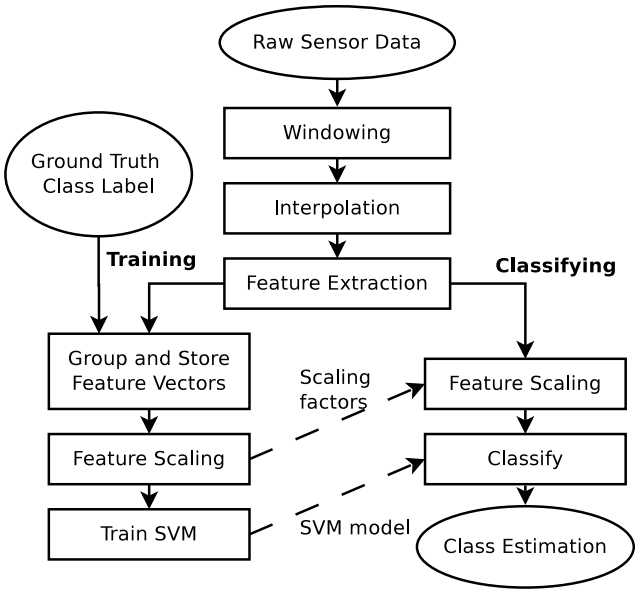


Figure 2: Flow chart of the classification algorithm

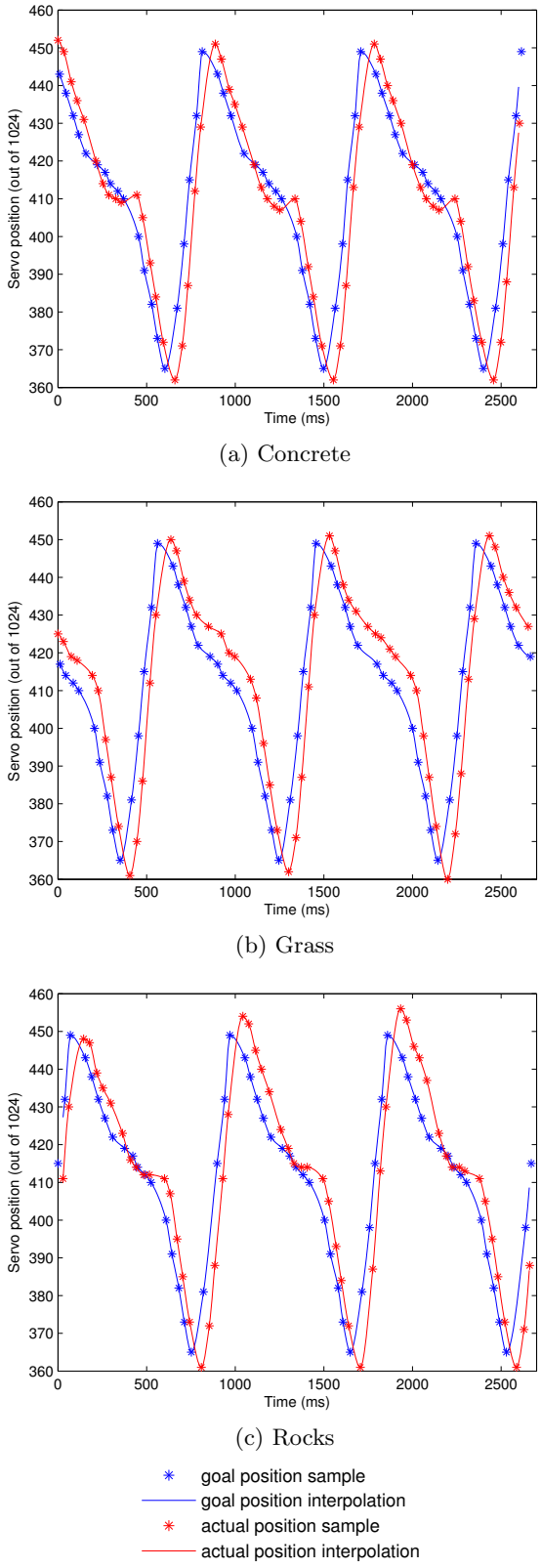
window. The figure shows subtle differences between the signals for the three terrains and our feature extraction algorithm is designed to exploit these differences. The sensor data is collected from the six servos on the front pair of legs and each signal is sampled at 20 Hz. We chose to use two legs (the front pair) for data collection rather than all six to increase the sample rate. There are other sensors built into the smart servos but we focus on the position sensors since this data alone produced valid results.

3.2 Windowing

The algorithm gives a new class prediction every 0.9 s (1 gait cycle) and each prediction is based on the past 2.7 s (3 gait cycles) worth of data. We chose 3 gait cycle length windows so the decision is more robust to variations in the terrain or short signal errors but still allows the algorithm to react quickly to changes in the terrain type. A uniform window is used at this stage while a Hamming window is applied later for the frequency domain features.

3.3 Interpolation

Data interpolation is required for several reasons: to estimate parts of the signal that may be missed due to the relatively low sample rate of the raw signals (approximately 20 Hz), and for synchronising the data between the different signals and giving a uniform sample rate suitable for the FFT used in the feature extraction. A cubic Hermite spline interpolation method fits piecewise cubic functions over each sensor signal. This method outputs a reconstructed function which is continuous and has a continuous first derivative. This is more appropriate than linear interpolation as it gives a more smooth and



(a) Concrete

(b) Grass

(c) Rocks

Figure 3: Data from the front left tibia over one 2.7 s window while traversing 3 different terrains. Subtle differences can be seen between the actual position signals for the three classes, for example at goal position = 415.

realistic reconstruction of the actual servo movements. The interpolation reconstruction is illustrated in Figure 3 for the front left tibia servo. This interpolation is applied to all signals and these interpolation functions are resampled at 100 Hz. The new time stamps are matched for all signals within a window so now the signals can be synchronised with each other. The reconstructed signals are then passed to the feature extraction stage of the algorithm.

3.4 Feature Extraction

The features focus on the servo control loop error by using the difference between the goal position and the actual position for each servo. Features are extracted from the error signal in both gait-phase and frequency domain to give a 660-dimensional feature vector.

Gait-phase domain

The main feature set exploits the cyclic nature of the leg movements. It achieves this by first transforming the interpolated position time series signals (Figure 3) into gait-phase domain (Figure 4). Features are then extracted from small subwindows of the gait-phase domain. The motivation for this method is that differences are observed between the terrain classes in certain sections of the gait cycle.

The transformation from time to gait-phase domain is achieved by processing the goal position signal of the front left coxa servo. This servo is the closest to the body and is responsible for the forward/backward horizontal motion of the robot. For a standard tripod gait, this servo moves in a triangle wave with the same frequency as the gait cycle. The goal position signal is used since it is independent from the terrain interactions, unlike the actual positions. The local minimum for each cycle of the goal position is mapped to a gait-phase value of 0. This phase closely corresponds to the times when the front left foot first touches the ground within each gait cycle. All other data points are given a phase value proportional to the time since the previous minimum. The exact same phase transformation values (based on the front left coxa) are used for all of the servos since all servos move with the same gait period.

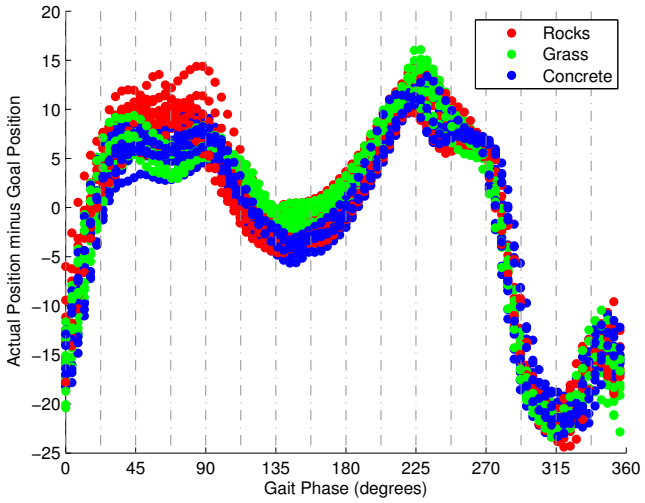


Figure 4: Gait-phase domain for three terrains over 12 steps for the front left tibia servo. Vertical lines show the 16 subwindow divisions used in the feature extraction algorithm. The foot is touching the ground between approximately phase 0 and 180.

Next, the control loop error signals are calculated for each servo by subtracting the goal position signals from the actual positions. Figure 4 illustrates the resulting phase domain transform for the front left tibia servo across three terrains. In this new gait-phase domain, the differences between the classes are more evident than in the original time domain.

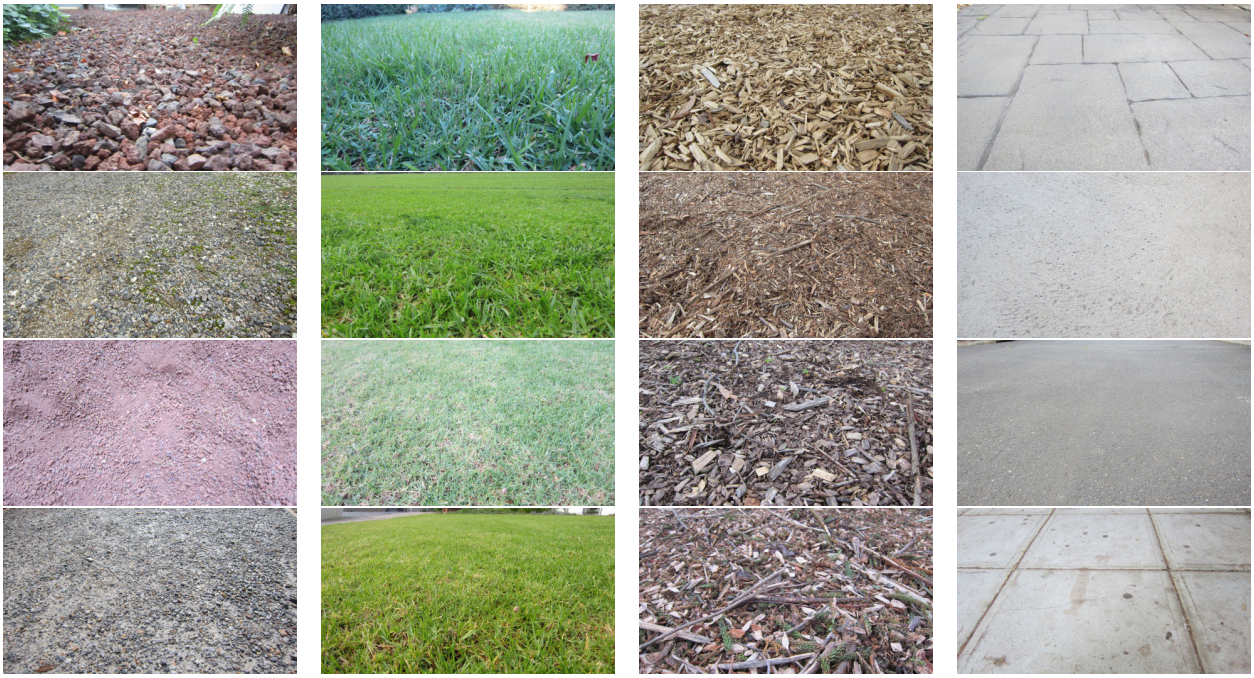
For the feature extraction, the gait-phase domain is evenly divided into 16 subwindows so that each subwindow contains a small segment of the gait cycle. The subwindow divisions are shown as vertical dashed lines in Figure 4. The algorithm calculates the minimum, maximum, mean, median and standard deviation of the error signal data points within each of the 16 subwindows. This is repeated for the six servos to give a total of 480 gait-phase features.

Frequency domain

The frequency domain features are calculated from the same interpolated position error signals in the previous feature set. The signals are transformed from the original time domain to the frequency domain by first applying a Hamming window then computing the discrete Fourier transform. Several features are then extracted from the frequency spectrum amplitudes for the first 25 bins up to approximately 12 Hz. Firstly, the raw amplitude values are used as features (25 times 6 servos gives 150 features). The centroid, standard deviation, skewness and kurtosis which measure the shape of the spectrum gives 24 more features. Finally, the energy is calculated as the sum of the squares of the amplitudes giving 6 features.

3.5 Feature Scaling

It is important to scale the features before using them in a SVM to avoid features in larger numeric ranges dominating features in small numeric ranges. Our algorithm linearly scales the training feature vectors such that each feature has a mean of 0 and standard deviation of 1. The same feature scaling factors that were used for the training vectors are applied to all new observations before being classified by the SVM. In this approach, it is assumed that all of the features have the same level of prominence. However, an alternative approach is to apply feature saliency detection prior to classification to



(a) Rocks (b) Grass (c) Mulch (d) Concrete

Figure 5: Test terrains grouped into their associated classes. These terrains were used for training and some of the testing of the classification algorithm.

devote more weights towards features that are more discriminative between the classes [Moghadam *et al.*, 2011].

3.6 Classification

The resulting feature vectors are input into a Support Vector Machine (SVM) classifier. In our offline implementation we use the built-in MATLAB SVM library and for the online implementation we use the open-source LIBSVM library [Chang and Lin, 2011]. We choose the linear kernel function for improved run time, since we did not observe any substantial performance improvements using higher dimensional kernels. We use the "one-against-one" approach to extend the binary SVM algorithm to multi-class classification.

4 Testing and Results

The datasets used in the tests were obtained from about 25 different outdoor locations with approximately 2-3 min worth of data (split into 4 runs) from each location. These datasets are grouped into *rocks*, *grass*, *mulch* and *concrete* classes, as shown in Figure 5. All of these datasets were recorded on flat sections of outdoor terrain. Training with data from multiple locations for each class means the classification model is much more likely to generalise to new locations, which is important for a robot when learning about a new environment. If all of the data within a class is from the same location, particularly within a controlled laboratory environment, then it is likely the classification model will overfit characteristics specific to that location.

An additional class (*free*) was created from data collected while the robot's legs were hanging above the ground. This class is useful for initial testing because a classifier should be able to make a clear separation between this class and the terrain classes. This is also useful for testing the online implementation since you can determine if this class has been trained correctly by simply picking up the robot.

4.1 Cross Validation

k-fold cross validation is one of the most common testing methods for classification algorithms. This method involves partitioning the class datasets into k random partitions. Classification is performed using one of the k partitions as the test set while the remaining $k-1$ partitions are used for training the classifier. This process is repeated k times such that each partition is used for testing exactly once. The performance can then be evaluated by adding the results from each of the k tests.

We use confusion matrices and precision and recall metrics to evaluate our results. In the confusion matrices, the columns represent the actual classes and rows represent the predicted classes. The first test, shown in Table 1, is a k-fold cross validation with $k = 10$ and using all data from the five classes. The confusion matrix shows

Table 1: Confusion Matrix of k-fold cross validation, with $k = 10$

Terrain	Rocks	Grass	Mulch	Concrete	Free	Precision %
Rocks	699	0	15	0	0	97.9
Grass	0	764	1	0	0	99.9
Mulch	14	0	462	0	0	97.1
Concrete	0	0	0	963	0	100.0
Free	0	0	0	0	423	100.0
Recall %	98.0	100.0	96.7	100.0	100.0	

Table 2: Confusion Matrix of leave-one-dataset-out cross validation

Terrain	Rocks	Grass	Mulch	Concrete	Free	Precision %
Rocks	388	12	111	5	0	75.2
Grass	5	732	2	0	0	99.0
Mulch	282	20	365	0	0	54.7
Concrete	38	0	0	958	0	96.2
Free	0	0	0	0	423	100.0
Recall %	54.4	95.8	76.4	99.5	100.0	

Table 3: Confusion Matrix of leave-one-dataset-out cross validation with combined *rocks* and *mulch* class

Terrain	Rocks and Mulch	Grass	Concrete	Free	Precision %
Rocks and Mulch	1144	36	5	0	96.5
Grass	7	728	0	0	99.0
Concrete	40	0	958	0	96.0
Free	0	0	0	423	100.0
Recall %	96.5	95.3	99.5	100.0	

precision and recall are above 95% for all five classes. Particularly high accuracy is seen for the *grass* and *concrete* terrains but there is some confusion between *rocks* and *mulch*.

4.2 New Terrains

One issue with the k-fold validation tests is that the training set always contains some data from the same location as the test set. To rectify this issue, the *leave-one-dataset-out* test partitions the data such that the training and test sets never contain data from the same location. This is achieved by using the datasets from one particular location for testing and using all other datasets for training. This is repeated such that each dataset is used for testing exactly once. Results from this test are shown in Table 2 and shows similar performance for the *grass* and *concrete* terrains.

However, there is significant confusion between *rocks* and *mulch*. The confusion indicates the robot's control loop is similarly affected by these two terrains since they have similar texture. Therefore the robot is not likely to need the ability to discriminate between these terrains and instead they can be combined into a single class. Table 3 shows the results of the same *leave-one-dataset-out* test but with a combined *rocks* and *mulch* class and shows precision and recall above 95% for the four classes.

4.3 Overlapping Terrains

Figure 6 shows the classification results while the robot moved across three different terrains in a natural outdoor environment. For this test, the five training classes were made up of all of the previously used datasets. None of these training datasets were recorded at the same location as where this test took place and therefore shows how the algorithm can generalise to new locations. The algorithm calculated correct classifications for the *grass* patches at the beginning and near the end of the test. The *rocks* and then *mulch* patches were classified correctly most of the time but with some mistakes. We expected some confusion between the classes since some patches consisted of a mixture or overlap of terrain types.

Figure 6: Terrain classification results by the ROS implementation for the robot walking (from bottom to top) over three overlapping terrain types. The false colour overlay represents the estimated terrain classes with the following colour scheme: green = *grass*, blue = *rocks*, red = *mulch*.

4.4 Execution Time

We measured the run time using the ROS online implementation running on a standard laptop computer with Ubuntu 12.04 64-bit, Intel Core i7 Quad CPU and 8 GB RAM. Measurements were taken from the time when a new 2.7 s window of data is ready to when this window has finished being classified. The average run time is 10 ms, with the feature extraction taking 8 ms and the SVM classification taking 2 ms. The CPU usage while running the software was less than 2 % and the RAM usage approximately 150 MB. Our initial intentions were to decrease the run time by reducing the dimensions of the feature space using a feature selection method. However, since the run time is considerably faster than the window intervals, we concluded this was not necessary.

5 Conclusions

We have presented a novel approach for terrain classification by a hexapod robot which utilises position measurements of the leg servos. The key advantage is that the position sensors are built into the actuators and therefore no additional sensors are required. Our approach relies on and measures aspects of the robot-terrain interactions. Therefore it provides the robot with insight into how its motion is being affected by the environment, which is important for an adaptive gait control loop. This is an advantage over exteroceptive sensing techniques, such as vision based, which can not provide this information. The ROS implementation demonstrates our classification algorithm is capable of performing terrain classification in real time. Testing shows our approach can successfully differentiate between *grass*, *rocks*, *mulch*, *concrete* and *free*. Some confusion was experienced between the *mulch* and *rocks* terrains due to the similar textures in these datasets causing similar robot-terrain interactions. If a robot requires better differentiation between *mulch* and *rocks* then an extension to our work would be to complement our approach with alternative sensor modalities for increased robustness and also to investigate different gait patterns with the aim of increasing the distance between the classes.

Acknowledgments

The authors would like to thank Marcus Hörger from CSIRO for writing some of the software for the hexapod. The authors gratefully acknowledge funding of the project by CSIRO SSN TCP.

References

[Belter and Skrzypczynski, 2011] Dominik Belter and Piotr Skrzypczynski. Rough terrain mapping and classification for foothold selection in a walking robot. *Journal of Field Robotics*, 28(4):497–528, 2011.

[Brooks and Iagnemma, 2012] Christopher Brooks and Karl Iagnemma. Self-supervised terrain classification for planetary surface exploration rovers. *Journal of Field Robotics*, 29(3):445–468, 2012.

[Chang and Lin, 2011] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27:1–27:27, 2011.

[DuPont *et al.*, 2008] Edmond DuPont, Carl Moore, Emmanuel Collins Jr., and Eric Coyle. Frequency response method for terrain classification in autonomous ground vehicles. *Autonomous Robots*, 24(4):337–347, 2008.

[Filitchkin and Byl, 2012] Paul Filitchkin and Katie Byl. Feature-based terrain classification for littledog. In

IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 1387–1392, 2012.

[Giguère *et al.*, 2006] Philippe Giguère, Gregory Dudek, Shane Saunderson, and Chris Prahacs. Environment identification for a running robot using inertial and actuator cues. In *Robotics: Science and Systems*. The MIT Press, 2006.

[Höpflinger *et al.*, 2010] Mark Höpflinger, David Remy, Marco Hutter, Luciano Spinello, and Roland Siegwart. Haptic terrain classification for legged robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2828–2833. IEEE, 2010.

[Larson *et al.*, 2005] Amy Larson, Guleser Demir, and Richard Voyles. Terrain classification using weakly-structured vehicle/terrain interaction. *Autonomous Robots*, 19(1):41–52, 2005.

[Lu *et al.*, 2009] Liang Lu, Camilo Ordonez, Emmanuel Collins Jr., and Edmond DuPont. Terrain surface classification for autonomous ground vehicles using a 2d laser stripe-based structured light sensor. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2174–2181, 2009.

[Moghadam and Wijesoma, 2009] Peyman Moghadam and Wijerupage Sardha Wijesoma. Online, self-supervised vision-based terrain classification in unstructured environments. In *IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pages 3100–3105, 2009.

[Moghadam *et al.*, 2010] Peyman Moghadam, Wijerupage Sardha Wijesoma, and MDP Moratuwage. Towards a fully-autonomous vision-based vehicle navigation system in outdoor environments. In *11th International Conference on Control Automation Robotics & Vision (ICARCV)*, pages 597–602, 2010.

[Moghadam *et al.*, 2011] Peyman Moghadam, Saba Salehi, and Sardha Wijesoma, Wijerupage. Computationally efficient navigation system for unmanned ground vehicles. In *IEEE Conference on Technologies for Practical Robot Applications (TePRA)*, pages 133–138, 2011.

[Ojeda *et al.*, 2006] Lauro Ojeda, Johann Borenstein, Gary Witus, and Robert Karlsen. Terrain characterization and classification with a mobile robot. *Journal of Field Robotics*, 23(2):103–122, 2006.

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

[Saitoh and Kuroda, 2010] Teppei Saitoh and Yoji Kuroda. Online road surface analysis using laser remission value in urban environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 15–21, 2010.

[Schmidt and Walas, 2013] Adam Schmidt and Krzysztof Walas. The classification of the terrain by a hexapod robot. In *Proceedings of the 8th International Conference on Computer Recognition Systems (CORES)*, pages 825–833, 2013.

[Vernaza *et al.*, 2008] Paul Vernaza, Ben Taskar, and Daniel Lee. Online, self-supervised terrain classification via discriminatively trained submodular markov random fields. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2750–2757, 2008.

[Wang *et al.*, 2012a] Shifeng Wang, Sarath Kodagoda, and Rami Khushaba. Towards speed-independent road-type classification. In *12th International Conference on Control Automation Robotics & Vision (ICARCV)*, pages 614–619, 2012.

[Wang *et al.*, 2012b] Shifeng Wang, Sarath Kodagoda, and Ravindra Ranasinghe. Road terrain type classification based on laser measurement system data. In *Proceedings of Australasian Conference on Robotics and Automation*, Wellington, New Zealand, 2012.

[Weiss *et al.*, 2006] Christian Weiss, Holger Fröhlich, and Andreas Zell. Vibration-based terrain classification using support vector machines. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4429–4434, 2006.