Milestone Report

Eric Trautmann
COSC 134

May 12, 2009

1 Introduction

Typical obstacle-recognition systems on mobile robots use stereo-vision or laser ranging to perform classification. The Yeti Robot, a robot designed to perform medium range autonomous traverses in Antarctica and Greenland, is not currently equipped with vision or LADAR, but has a proprioceptive sensor suite including 6-axis IMU, wheel encoders, wheel motor current, and GPS. The purpose of this project is to increase the mobility of the Yeti robot by enabling it to recognize obstacles in real time, so it can appropriately recover in the event that it becomes immobilized. Different discrete types of obstacles, like a step obstruction, steep slope, or deep snow, require different control strategies for optimal recovery. By identifying how the robot is immobilized, the robot can then apply an appropriate recovery strategy.

2 Methods and Tools

For this project, I'm using Hidden Markov Models to perform sequence classification on sensor data collected using the robot. HMMs are frequently used for different sequence classification tasks, including handwriting recognition, speech recognition, or gesture recognition [1, 2, 3].

HMM's are used as a model to represent the probability of state transitions between the 'hidden' system states. Each collected point of sensor data represents an observation, and each hidden state has some probability of producing a given observation. States can either produce discrete outputs, or real-valued outputs with some probability distribution. Since the sensor

data collected here is real-valued, it makes sense to use an HMM that models outputs as a gaussian or mixture of gaussians (MOG). The alternative, if discrete output values are preferred, is to use a clustering method to group vectors of sensor data into discrete values. This suggests a degree of data compression prior to classification. As a result, we have chosen to pursue an implementation using gaussian (not MOG) outputs.

There are three canonical problems associated with HMM's [2] [4].

- Problem 1 - Given a sequence of observations $O_1, O_2, \ldots, O_t$ and a trained HMM, find the probability this model produced the output sequence observed.
- Problem 2 - Given an observation sequence, find the optimal state sequence that produced these observations.
- Problem 3 - How can we create a model λ given a series of observations in order to maximize $p(O, \lambda)$?

2.1 Methods

These three canonical problems have been 'solved' and standard algorithms exist to address each one. This work seeks to verify whether or not it is possible to perform obstacle classification using proprioceptive sensor data to train HMMs. The proposed classification method works as follows:

1. Collect sequences of data during multiple immobilization events over different types of obstacles
2. Train one HMM for each obstacle using multiple sequences of observations using the Baum-Welch Algorithm.
3. Classify new sequences by computing the probability that each obstacle classifier produced this sequence of data. The model with the highest probability of producing the sequence is chosen. This procedure uses the forward-backwards algorithm.

In reality, the number of datasets available for classification is extremely limited. Ideally, we would like to compare data from three main classes of obstacles that the robot faces in a polar environment: step obstructions (sastrugi features), steep slopes, and deep snow. Unfortunately, not enough data

was available for deep snow to be able to perform meaningful classification, so the validity of this method must be tested using step obstructions and slopes only.

Collecting data is extremely expensive, as each run can take up to an hour to collect and process. Nine runs worth of data are available for each feature. Data collection is discussed in more detail in section **??**. Due to the small amount of training data, classification is performed by holding out one run from a set of features. An HMM for this obstacle is trained using the other eight runs, and the model representing the other obstacle is trained using all nine available data runs. The probability that each model produced the held-out sequence is computed, and the sequence classified based on which model is more probable. This process is repeated for all eighteen data runs to produce a final classification score.

It is important to note the differences between this procedure and a more classical model selection problem. In this case, the models used for classification are not re-used between sequences. This is not the same procedure as leave-one-out cross validation, in which cross validation is used to select the best model. In this case, we're not trying to optimize model performance as a function of parameters, but instead want to gauge classification accuracy using a limited set of data.

2.2 Tools

This project has been implemented using data collected with the Yeti robot. Data were collected during runs in which the robot was driven manually while sensor data was logged using a CR3000 Datalogger. The collection of runs was imported into matlab, and processed into datasets using the procedure outlined in section **??**.

The two major components of this project are: 1) data acquisition and data set preparation and 2) data processing and classification. Matlab is used to import, filter, scale data, and process data into cell arrays that represent a complete set of data for all runs. To date, the Baum-Welch algorithm and forward-backward algorithms are implemented using an open-source HMM toolbox available under the GPL[5].

3 Data

Data were collected for nine runs in which the robot became immobilized on different step obstructions, and nine runs in which the robot was immobilized after it failed to climb a slope. Data were collected on two different days and on different steps and slopes. Six of the slope obstacle runs and three of the step obstruction runs were collected on 2/25/09 on man-made obstacles at the rugby fields north of campus, and the remainder of the runs were collected on 3/25/09 at a different field further north.

Conditions on 2/25/09 were below freezing with densely packed snow that was several days old. On 3/25/09, the temperature was above freezing, and the snow was heavy, wet, and slushy. This made it difficult for the robot to gain traction, which is perfectly acceptable for these tests. Due to the difference in surface conditions, we would expect the signature from the IMU, the distribution of motor currents, and other features included in the feature vector to vary significantly between these two days.

Data were imported into Matlab and inspected to ensure that each of the sensors was working properly during each run. Runs in which sensors were not working or outputting spurious data were culled before adding data to the data set. Each of the nine runs collected for each type of obstacle represents a 'clean' data run with no spurious observations. It is an interesting matter for further study to consider the impact of false sensor data on classification, but has not yet been considered in this work.

Each sequence from a data run is intended to capture the class of the obstacle using data from immediately before immobilization. For simplicity, we consider each data run to be approximately two seconds worth of data preceding immobilization. This corresponds to a sequence length of 41 data points in each run. The exact indices of which data to use were selected by hand using a simple threshold scheme to define full immobilization. The end of a data sequence was tagged at five samples past the point where each wheel exhibited wheel-slip above 90%. Wheels slip for each wheel is defined as:

$$S = \frac{(R\omega - V)}{R\omega} \tag{1}$$

where R is the wheel radius, V is the longitudinal velocity of the wheel center, approximated by the optical velocity sensor, and ω is the angular speed of the wheels as calculated using differential measurements from the

wheel encoders.

The data sets were constructed using all possible data that *might* be used for HMM-based classification. The classification work performed in section 4 does not use all of these features. Future work may incorporate a combination of features from this set, but it is unlikely that all features will be necessary to perform classification.

Each input feature is filtered using a moving-average filter with a window of 5 samples to perform basic smoothing. A more sophisticated digital filter could easily be implemented, but has not yet been implemented. Data are filtered on the original data set prior to selecting the 41-sample range for one set. This ensure that all data in a given run are filtered identically, and the data at the beginning of the sequence are calculated using data values from several points outside the range. This is, of course, perfectly valid whether we choose to perform sequence classification on or off-line.

The data sets are presented in figures 2-5. These figures are produced as images in Matlab, which helps one to 'see' the sequence and the relationship between features in the data. Figure 1 provides a key for interpreting each data sequence image. The images use pixel values scaled relative to the data within each image, so the same color does not necessarily represent the same data value in two different images.

From these data visualizations, we can see that clear patterns in the step and slope obstacles, suggesting at a qualitative level that sequence classification is possible. The bottom two rows in each image are the acceleration derived from differentiating optical velocity measurements, and the Z-axis acceleration from the IMU. We see that these two features are fairly erratic, and may decrease classification performance.

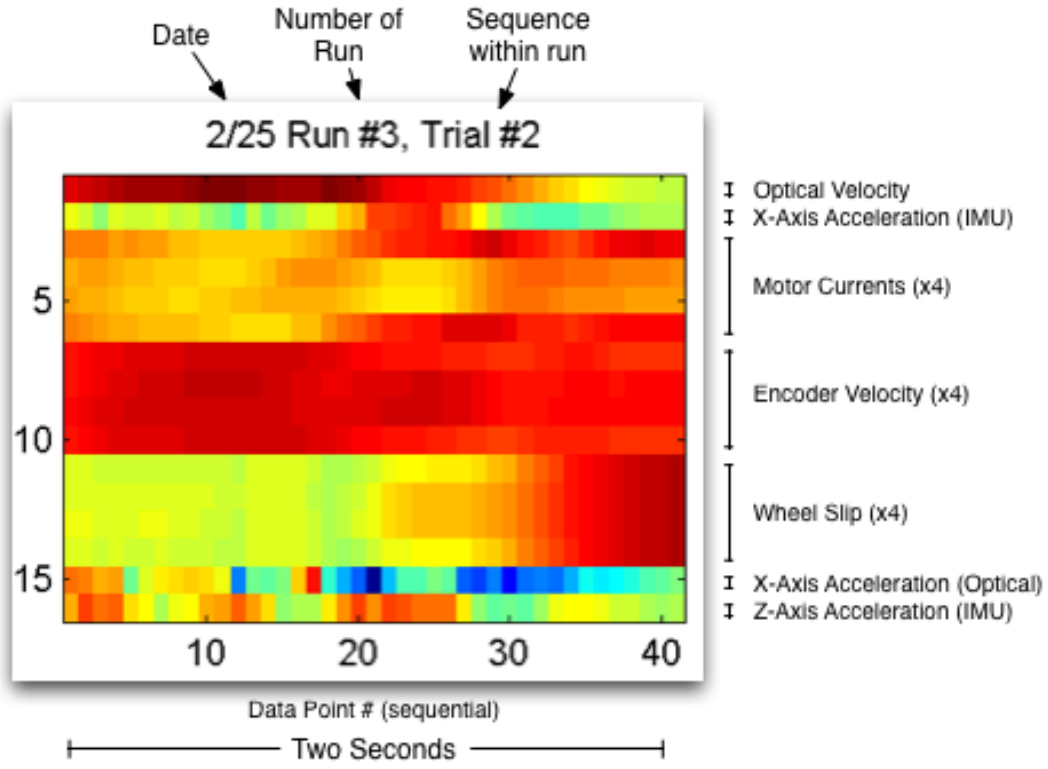

Figure 1: **Example Data image** - The purpose of imaging data this way is to observe the relationships between features for each type of obstacle. Colors are relative to each individual image and not representative of absolute data values.

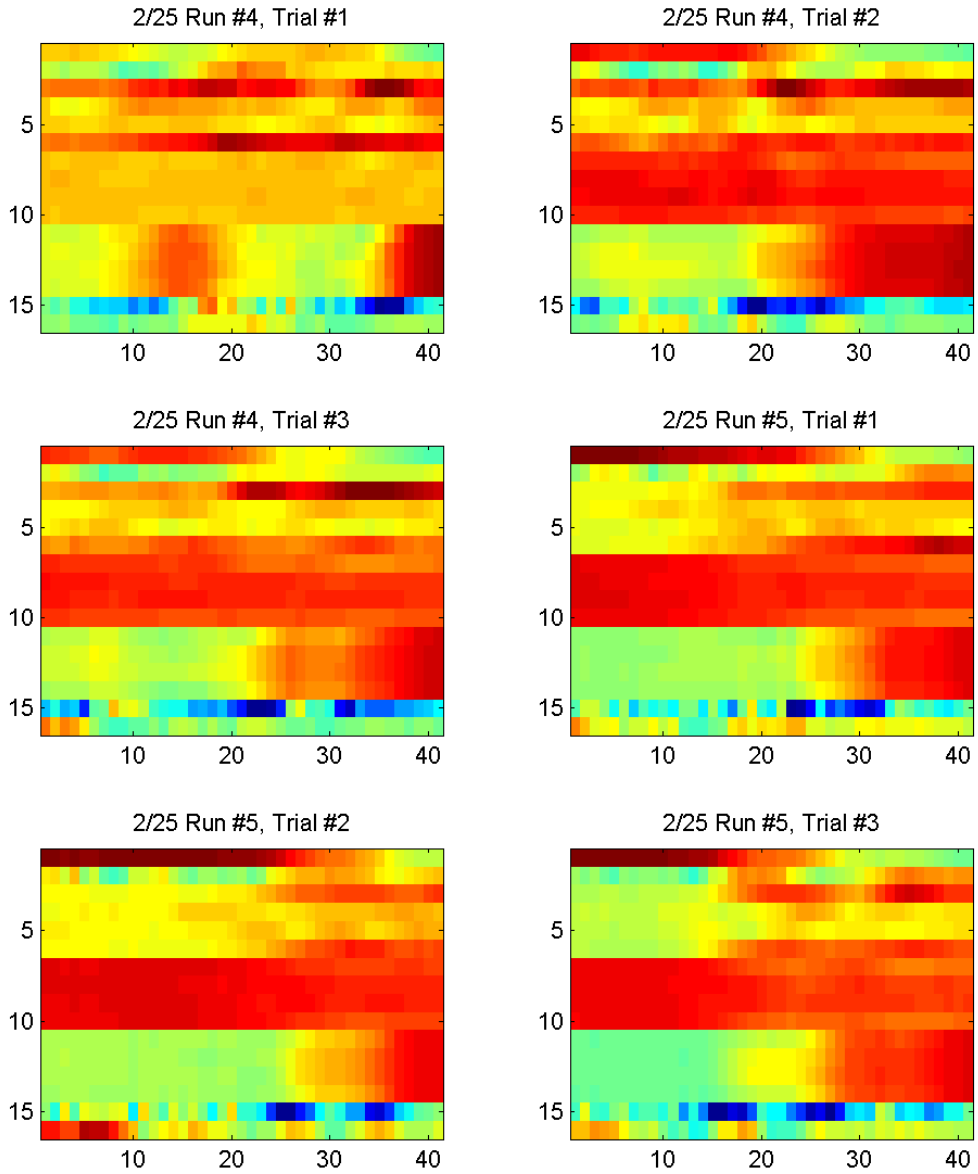

Figure 2: Immobilization events recorded on slope obstacles from 2/25/09.

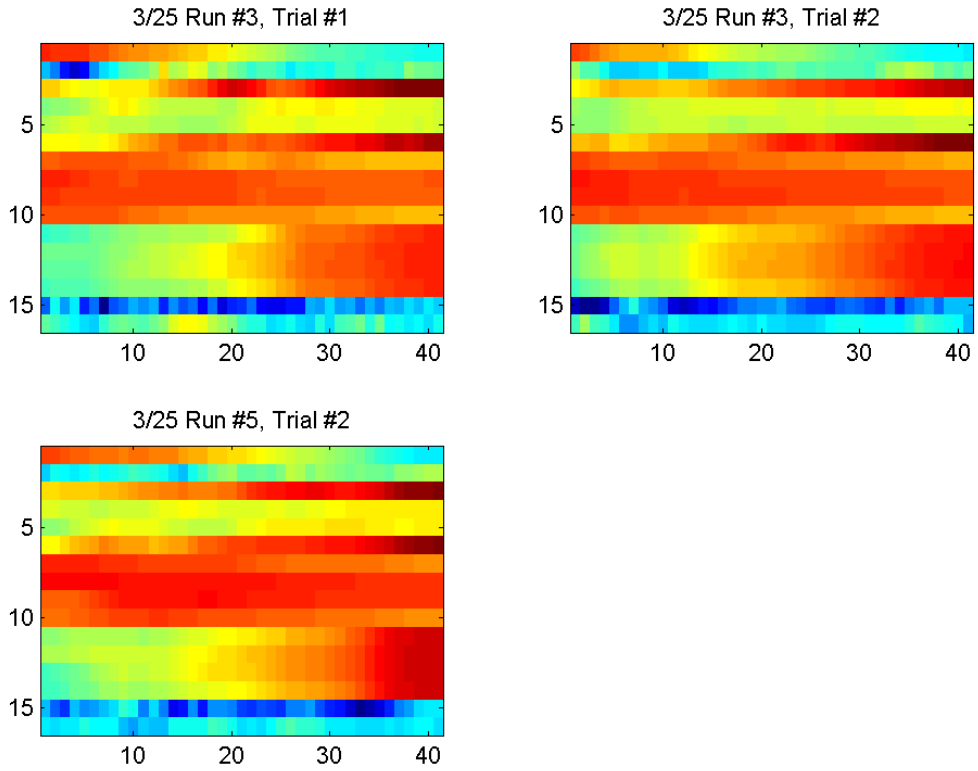

Figure 3: Immobilization events recorded on slope obstacles from 3/25/09.

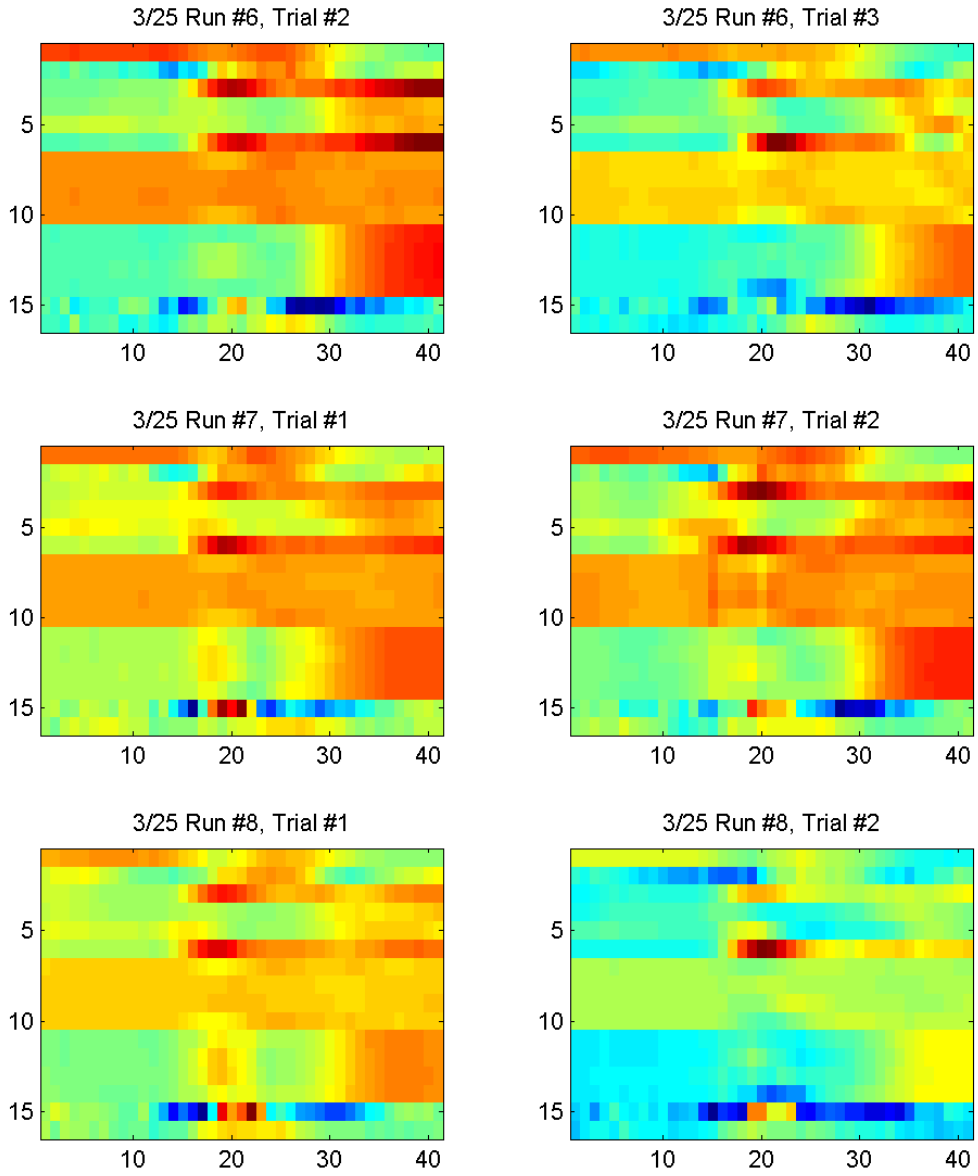

Figure 4: Immobilization events recorded on step obstacles from 3/25/09.

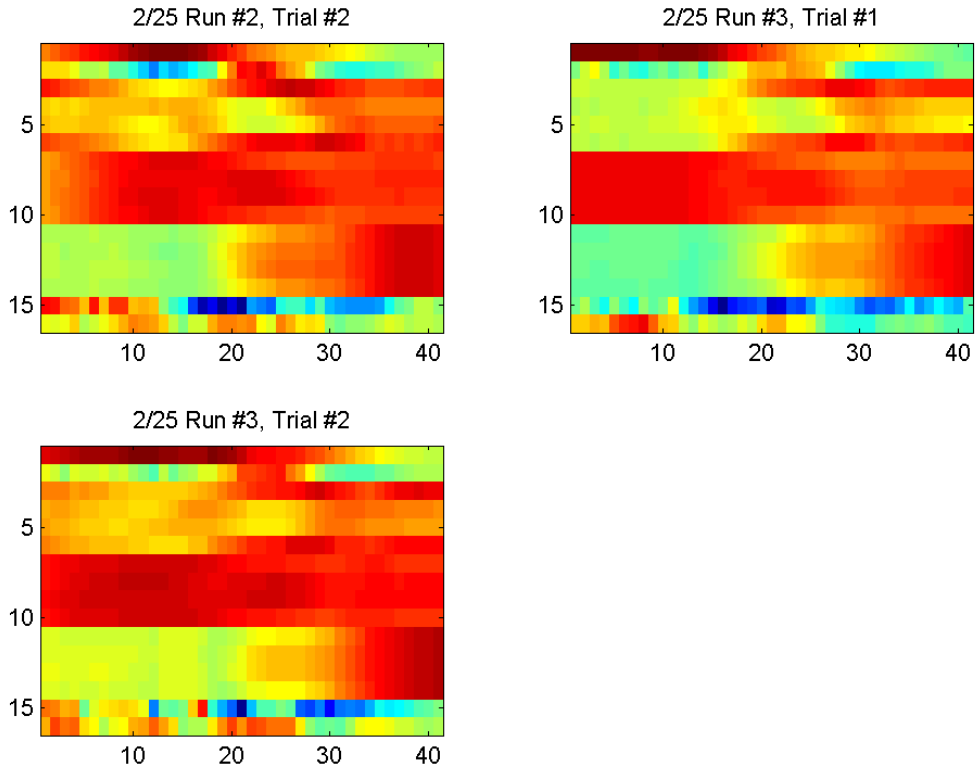

Figure 5: Immobilization events recorded on slope obstacles from 2/25/09.

4 Hidden Markov Model Sequence Classification

Several test were conducted using different combinations of data runs, feature vectors (subsets of the data presented above), and model parameters.

4.1 Test 1

The first test was performed using a data set consisting of the six slope obstacle trials collected on 2/25 and the six step obstruction trials collected on 3/25/09. These runs are the most similar within a given obstacle type, so classification performance should be the best under these conditions.

The test was conducted according to the method outlined above. Each trial was classified by constructing an HMM using all other trials for that obstacle, while a second model was created using all six of the obstacles for the opposite class. HMM's were constructed using an arbitrary value of six hidden states, though we have little intuition to gauge what a state actually represents in this case. In future tests, I will use cross-validation to select an optimal number of hidden states.

The resulting class was selected based on which model yielded the highest probability of creating the observation sequence in the trial that was held out during training. This process was then repeated for each of the 12 trials in the test. Only six features were used, including optical velocity, IMU X-axis acceleration, and four-wheel motor currents. Intuitively, we would expect that these features would contain the greatest signal. The robot experiences different levels of deceleration as it contacts a step vs. a slope, and once immobilized, the robot's weight distribution and contact angle likely leads to differences in the distribution of motor currents among the four wheels for a given obstacle. Principal component analysis will be used to analyze features in more depth in the future, but has not been completed yet.

For this limited feature set, classification performance was extremely high. For each of the twelve trial runs, the model corresponding to the correct obstacle type yielded a higher probability, and the overall classification accuracy was 1. This high accuracy may be due to overfitting in the sense that the limited data sets used to create these models represent small variation in obstacles within a given class. In this case, data for each obstacle were collected from a similarly shaped feature on one type of snow. Test 2 addresses these

concerns by using more data.

Table 1: Test 1 - Confusion Matrix

Predictions\Targets	Step	Slope
Step	6	0
Slope	0	6

4.2 Test 2

Test two was performed using all nine trials for each obstacle type. The HMM's were trained using the assumption of six hidden states and the same test procedure as in test 1. This test resulted in a classification accuracy of 88.9%. Out of eighteen examples, two of the step obstructions were misclassified as slopes.

Table 2: Test 1 - Confusion Matrix

Predictions\Targets	Step	Slope
Step	7	0
Slope	2	9

5 Discussion

These results, while based on extremely limited data sets, support the theory that HMMs can be used to perform obstacle recognition using proprioceptive sensor data. It is clear that a system intended to be used online in a field-deployed robot must be trained using more extensive data than available for this work. Despite this, from these preliminary results, we have strong reason to suspect that this method works and can be generalized to other types of obstacles.

These preliminary results suggest that we need additional data on different surface conditions and different obstacles to more accurately gauge

the performance of an HMM-based obstacle recognition classifier. Unfortunately, it will not be possible to collect more data since snow surfaces are not available during the spring.

Finally, these results also suggest that it may be possible to use the same methods for other questions of interest to the robotics community, such as terrain classification. Current terrain classification methods typically use visual classifiers or calculations of wheel force vs. wheel slip to calculate terrain parameters. Using HMMs for terrain classification would require sensitivity to higher-frequencies, and thus a higher sampling rate.

6 Project Stage Completion

So far, the project is on track. This work validates the use of HMMs for obstacle recognition, and a framework of code has been created to train models and classify sequences. In the proposal document, I outlined a plan to compute state sequences given a sequence of observations (HMM problem 2). This has not yet been completed, but is low priority as it is not necessary for classification and thus validation of the current method.

There are still several remaining open issues which have not been solved. First, the calculated log-likelihood when training a new model is often larger than one, suggesting that the covariance is shrinking to a point. I have not yet looked into this in detail, and will address this in future work.

Second, the state transition matrices between states sometimes contain absorbing states, where transition probability is 1 for the present state and 0 everywhere else. This is unlikely to represent a real physical system, and needs to be looked at in greater depth.

In addition, I would like to test a range of different numbers of hidden states. Classification performance is high given the current arbitrary value of six states, but the computational cost of the entire procedure would decrease if fewer states are used. Finally, PCA or some other procedure should be performed to select which features are most important for classification. If we can reduce the number of features used, the resulting classifier will be significantly more computationally efficient during classification.

References

[1] F Jelinek. Statistical methods for speech recognition. *books.google.com*, Jan 1998.

[2] L Rabiner and B Juang. An introduction to hidden markov models. *ieee assp magazine*, Jan 1986.

[3] J Bilmes. What hmms can do. *IEICE TRANSACTIONS on Information and Systems*, Jan 2006.

[4] R Dugad and U Desai. A tutorial on hidden markov models. *citeseerx.ist.psu.edu*, Jan 1996.

[5] Kevin Murphy. Hidden markov model (hmm) toolbox for matlab, 1998.