

Teaching Robots Where to Drive using Real World Data



Oliver Bartlett

Worcester College

Supervisor: Ingmar Posner

October 2019

Abstract

In this work we teach a robot where to drive from measured, real world data. In Chapter 1 we measure energy usage on a pedestrian robot and use high-level, publically available maps to predict energy consumption on unobserved routes with a heteroscedastic Gaussian Process model. These energy predictions can be used to make route-planning decisions in energy constrained situations, for example by learning to avoid areas with rough surfaces or that are likely to be crowded. Our method performs well, particularly in its ability to capture uncertainty in the data, however it is reliant on well annotated maps.

In Chapter 2 we consider the case where well annotated maps and good prior information about an environment are not available. Again we use measured, real world data by introducing vertical variance in a laser pointcloud as a proxy for drivability which can be labelled automatically in an online setting. We propose a method for using this online stream of data in a computationally efficient manner to learn where to drive. We then test this method against models trained in other environments and an existing method which utilises the most recent data.

Contents

1	High Level Route Planning Using Publicly Available Maps	5
1.1	Introduction	5
1.2	Related Work	7
1.2.1	Motivation	7
1.3	Technical Approach	8
1.3.1	Estimating noise-free power consumption	9
1.3.2	Modelling Empirical Noise	9
1.4	Experiments	10
1.4.1	Platform	11
1.4.2	Map Features	11
1.4.3	Data Collection	12
1.4.4	Sensor Data Post-Processing	12
1.5	Results	13
1.5.1	Energy Consumption Predictions via Heteroscedastic Regression	13
1.5.2	The Benchmark	14
1.5.3	Map Feature Ranking	17
1.6	Conclusions	17
2	Learning Terrain Drivability Online using Laser for Self-Supervision	18
2.1	Introduction	18
2.2	Related Work	19
2.2.1	Motivation	20
2.3	Framework Description	21
2.3.1	Using Laser Variance as a proxy for Drivability	21
2.3.2	Image to Drivability model	24

2.3.3	Online Global Image Descriptors	26
2.3.4	Experience Based Off-Road Algorithm	28
2.4	Experiments	31
2.4.1	Data Collection	31
2.4.2	Benchmarks	32
2.4.3	Metrics	33
2.5	Results	34
2.5.1	Preliminary	34
2.5.2	Accuracy	35
2.5.3	Algorithmic Efficiency	36
2.5.4	Timing	37
2.5.5	Discussion	41
2.6	Conclusions	42
3	Appendices	44

Introduction

This thesis outlines work performed to teach autonomous vehicles where to drive using real world data and experiences as explained below.

The framework presented in both chapters measures real world values which are indicative of drivability, learns a probabilistic model to relate relevant features to this measured drivability, and predicts the drivability in other locations using the model.

Rather than relying on expensive, subjective human labelling for data, we use the robot to automatically measure real world values which are indicative of drivability, in the same spirit as proprioception.

Using these measurements, as well as pertinent environmental features, statistical models can be learned which are able to predict drivability in locations before the robot traverses them, aiding navigation and path planning.

In Chapter 1, the energy consumption of a robotic platform is used as a proxy for drivability. These measurements are inherently non-stationary and noisy. However using features extracted from publicly available online maps, as well as a statistical model capable of modelling uncertainty, we are able to generalise energy consumption to untraversed routes similar to those already driven.

The developed algorithm is useful for making high level decisions about routes a robot should drive. For instance, would it be more efficient for a robot to drive a shorter route, but one which might be more crowded or have a rougher surface, or should the robot drive a longer distance on smoother, quieter roads?

We learned that using open source, online maps provided a rich feature set for learning a model which could predict drivability on untraversed paths. However, a major drawback of the algorithm is that the quality of annotations varied enormously between different locations.

When operating the robot in a different environment such as Cornbury Park Estate (Figure 1b), Open Street Maps has no information about the paths or terrain, and identifying path features is no longer possible.

Chapter 2 seeks to solve this problem by answering the following question - What is the most pertinent training data for a robot in a novel environment? Can it be taken from different datasets, in different environments or on different platforms?

To answer this, a large data collection exercise was undertaken with two different platforms in three different environments. Figure 3.2 (p 47) shows that the appearance of the different datasets is varied. There are no guarantees that models trained previously in one environment will perform well when used in a different environment.

Our analysis of driving in novel environments consists of two lines of enquiry: Firstly we utilise data collected in other environments to explore whether they are still effective. Secondly we utilise data collected by the robot as it navigates the environment. This includes testing an existing method and proposing a new method which is more computationally efficient.

With the exception of the Cornbury dataset in Chapter 2, all the data used in this project was collected by the author. The Literature Review is split between the two chapters as the prior art is different for each.

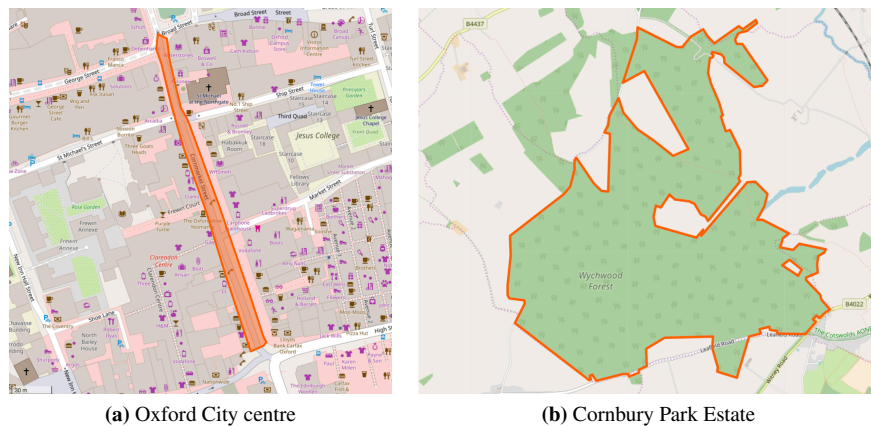


Figure 1: Difference in annotation quality between Oxford and Cornbury. Oxford area is public and annotated in great detail down to individual shops, whereas Cornbury is private land with no detailed annotation. Note that Cornbury map shows area 300x larger than Oxford City Centre.

Chapter 1

High Level Route Planning Using Publicly Available Maps

This Chapter is taken from the following paper:

O. Bartlett, C. Gurau, L. Marchegiani, and I. Posner, Enabling Intelligent Energy Management for Robots using Publicly Available Maps, in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, South Korea, 2016.

1.1 Introduction

Robots are limited by heavy and expensive batteries, which makes energy efficiency a key constraint on robot performance. Modelling and managing energy consumption is thus of vital importance to predict the lifetime and range of autonomous platforms. In this paper we propose a probabilistic, data-driven approach to estimating the energy consumption of a mobile robot on a set of trajectories, whether they have been traversed or not. In particular, we treat the robot as a *black box*, thereby removing our reliance on often unavailable system characteristics. Rather than evaluating the energy consumption due to the robot's locomotion and sensor hardware as well as computation costs, we measure consumption directly on the routes traversed and attempt to utilise features derived from publicly available maps to extrapolate to energy consumption on previously unseen routes. Figure 1.1 shows a typical output of our approach. In an urban environment our model predicts the energy consumption for a given set of path segments. Crucially, this information can be used by energy management systems or planning algorithms to forecast the attainability of a new goal given a particular trajectory.

Our approach is to frame this as a regression problem whereby *power* consumption is regressed on a feature representation of individual path segments. We elect to model *power* rather than energy consumption as it increases the robustness of our approach in two important ways: firstly, power estimates are more

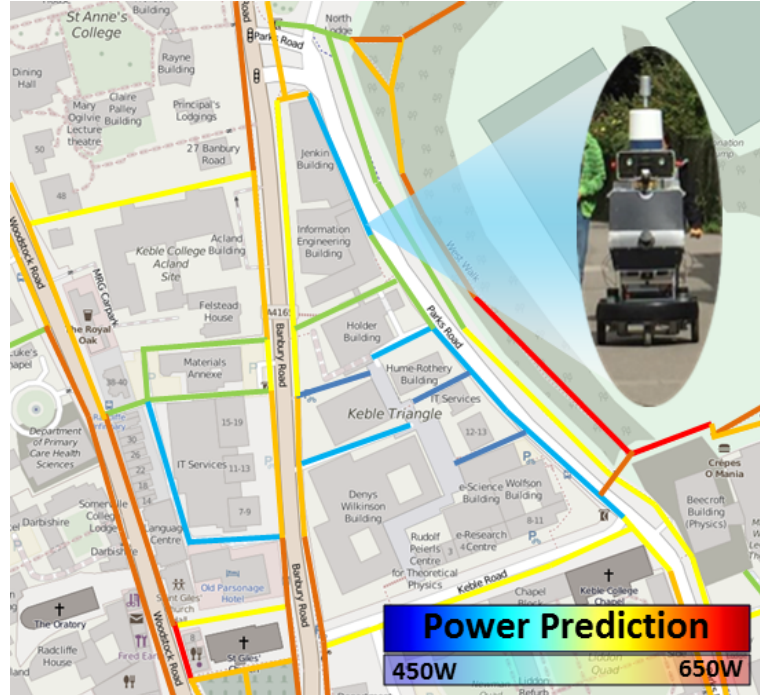


Figure 1.1: Example of a typical environment traversed by the robot. The output of our model shows the predicted energy consumption for each segments of our environment, whether visited by our robot or not. Note the high energy cost through gravelled park (East) and crowded retail area (South West).

robust to localisation errors. Consider a situation where the robot momentarily localises on an incorrect path segment. A small energy measurement on that segment will represent a significant outlier, the algorithm might perhaps record that only a few Joules were expended to traverse a significantly long stretch, whereas a power reading will still be within a sensible range as it is not dependent on the amount of time taken to traverse a particular segment, as an energy measurement would. Power estimates are also robust to unpredictable velocity profiles such as caused by traffic or waiting at a pedestrian crossing, they are not dependent on time taken to traverse a crossing, whether waiting a long time or short. We will demonstrate that our power consumption estimates can be converted into predictions of energy consumption by way of a simple velocity profile as commonly provided by a trajectory planner.

While power estimates are considerably more robust to errors in the time spent on a particular path segment, there nevertheless exists a substantial amount of variation due to unknown factors such as pedestrian traffic, weather conditions and different operator behaviours. A crowded environment, for example, may require the robot to manoeuvre to avoid people, thus increasing the variance in power consumption. An uncrowded and open pavement, on the other hand, will likely be traversed similarly each time resulting in a comparatively low variance. To account for this variability, we use the *most likely heteroscedastic Gaussian Process regression* framework proposed in [Kersting et al., 2007], which combines two separate Gaussian Processes (GPs) in order to explicitly accommodate non-uniform measurement noise.

To the best of our knowledge, this is the first work on predicting the energy consumption of a mobile

robot using only high-level, publicly available path features – thereby enabling predictions on unseen routes and trajectories.

The remainder of the chapter is structured as follows: Section 1.2 investigates the existing literature, in Section 1.3 we describe the theory of our approach, Sections 1.4 and 1.5 describe our experiments and results using 30km of data collected with a pedestrian robot. We further investigate the predictive power of the individual features within the model.

1.2 Related Work

Many different methods have been proposed to assess and reduce energy and power consumption in robotic platforms. Motion planning and optimisation of a robot’s speed profile are investigated in [Mei et al., 2004, Mei et al., 2006a, Kim and Kim, 2005]. The impact that processing and sensing have on power and energy consumption is analysed in [Mei et al., 2005, Fallah et al., 2013, Brateman et al., 2006]. The possibility of reducing energy consumption by systematically turning off the main localisation system is proposed in [Ondruska et al., 2015], [Wang et al.,] optimise sensor movement to achieve the same result. Generally, these approaches rely on the assumption that the power estimates are accurate. This work aims to fulfil this assumption by introducing more accurate techniques to model both the power consumption and the uncertainty of estimates. Model based approaches can be used to more accurately analyse the effective range of a robot. [Tiwari et al.,] use an analytical model of a robot’s component systems, [Hayes et al., 2011] use a simplified power train model while [Carroll, 2012] considers the effects of driving characteristics.

There are many different features which have been investigated that affect power and energy consumption. Mei et al. [Mei et al., 2006b] show that acceleration and deceleration substantially increase power requirements. The impact of specific terrain features is analysed in [Sun and Reif, 2003] and [Broderick et al., 2014]. Ishigami et al. in [Ishigami et al., 2011] consider terrain elevation and the steering and driving manoeuvres necessary to follow a particular path. Energy-efficient path planning based on estimating terrain traversability is investigated in [Martin and Corke, 2014].

1.2.1 Motivation

We can gather lots of data about places we choose to visit. Hundreds and thousands of terabytes of data have been collected in the form of laser point-clouds and camera images, to name the most common. These different sensor modalities potentially present a rich set of features to predict energy usage in mobile robots (for instance, the heights of curbs a robot might have to cross). *However*, these rich features are only available in places which we go and collect them (barring a few corner-case datasets such as KITTI). At this point it is just as easy to collect the energy usage data itself and not have to do prediction at all.

We need to be able to utilise other, less traditional features to inform our robot of its potential energy use in an environment. Such features would have to be available to us (or the public more generally), and a more widely available set of features would result in a more useful algorithm.

The features considered in our algorithm were Google Maps, Google Streetview and Open Street Maps. Google Maps provides aerial views of a location, and Google Streetview provides data taken from a ground vehicle travelling along roads and streets. Both might contain pertinent features such as surface type, or potentially how busy a place was (judging by numbers of cars and people). Open Street Maps (OSM) is an open-source mapping project maintained by collaborators in a Wikipedia style format. OSM offers information such as road surfaces, the surrounding shops and amenities, as well as information that is specific to a particular road segment (For instance, that the loading times for vehicles on Cornmarket are 18:00-10:00, or that the speed limit on the Champs-lyses is 50km/h).

We used OSM over Google Maps and Streetview because the pertinent information (such as surface type) had already been distilled by the map creators and annotators. Image based models are more complex, requiring more parameters, and it would have been possible to overfit such a model to a small training dataset. The small training dataset also motivated the use of models which were able to deal with uncertainty, such as Gaussian Processes.

The assumption is that this algorithm, once deployed, would be able to be integrated with a planner in order to come up with high level paths through an environment which were more energy efficient, by inferring from information available from OSM. This might be by verifying plans from more simplistic planners, or by incorporating them into a more complex, probabilistic planner.

1.3 Technical Approach

We define our routes as a collection of connected path segments. The path network and the characteristics of the segments are obtained from collaborative projects, such as OpenStreetMaps (OSM) [Haklay and Weber, 2008]. Each segment in the OSM map is defined by a feature vector $s \in \mathbb{R}^D$, containing information such as the number of intersections with other segments or surface type. A full explanation of chosen features is given in Section 1.4. Let p_i denote the power consumption while traversing segment s_i .

Following [Murphy et al., 2012] we implemented a heteroscedastic Gaussian Process [Kersting et al., 2007] as it was able to account for variability from latent factors such as traffic or weather conditions.

There are many different implementations of heteroscedastic probabilistic models, such as [Le and Smola, 2004], [Goldberg et al., 1997], [Miguel, 2011] [Binois et al., 2018]. However the approach taken by [Kersting et al., 2007] is a simple extension of the traditional Gaussian Processes framework, can be performed without Monte Carlo sampling yet is still able to capture the uncertainty of the power measure-

ments.

This uses two different GPs to model the power consumption and the variance of the measurements. The goal of the regression is to infer the posterior distribution $p(P^*|s^*, \mathcal{D})$ of using power P^* to traverse an arbitrary segment s^* , given a training set $\mathcal{D} = \{(s_i, p_i)\}_{i=1}^n$ of n power measurements $\mathbf{P} = (p_1, \dots, p_n)^T$ collected traversing training segments $\mathbf{s} = (s_1, \dots, s_n)^T$. An initial GP (GP₁) is used to estimate the noise-free power consumption. The empirical noise is modelled by a second GP (GP₂). A third GP (GP₃) is calculated which incorporates both the noise free estimates and the empirical noise. A convergence process then takes place which iterates until all empirical noise is accounted for. The approach is outlined in Algorithm 1.

1.3.1 Estimating noise-free power consumption

Our algorithm initially fits a standard homoscedastic GP to the training data:

$$P(\mathbf{s}) \sim \text{GP}_1(m_1(\mathbf{s}), k_1(\mathbf{s}, \mathbf{s}')) \quad (1.1)$$

where $m(\mathbf{s})$ and $k(\mathbf{s}, \mathbf{s}')$ are the mean and covariance functions chosen to reflect prior information about the data and to maximise the likelihood. GP₁ is used to predict the average noise-free power required to traverse $\mathbf{s}^* = (s_1, \dots, s_q)^T$, an arbitrary vector of q segments:

$$\mu_1(\mathbf{s}^*) = M^* + K^* K^{-1}(\mathbf{P} - M), \quad (1.2)$$

$$\Sigma_1(\mathbf{s}^*) = K^{**} - K^* K^{-1} K^{*T}. \quad (1.3)$$

In Equations 1.2 and 1.3, we have:

$$M \in \mathbb{R}^n, M_i = m_1(\mathbf{s}_i)$$

$$M^* \in \mathbb{R}^q, M_i = m_1(\mathbf{s}_i^*)$$

$$K^* \in \mathbb{R}^{q \times n}, K_{ij}^* = k_1(\mathbf{s}_i^*, \mathbf{s}_j)$$

$$K \in \mathbb{R}^{n \times n}, K_{ij} = k_1(\mathbf{s}_i, \mathbf{s}_j)$$

$$K^{**} \in \mathbb{R}^{q \times q}, K_{ij}^{**} = k_1(\mathbf{s}_i^*, \mathbf{s}_j^*)$$

1.3.2 Modelling Empirical Noise

After fitting the GP_1 , the residual noise r_i can be found on each segment s_i in the training segment set $\mathbf{s}$:

$$r_i = \frac{1}{n_i} \sum_{j=1}^{n_i} (P_j - \mu(s_i))^2 \quad (1.4)$$

where n_i is the number of power observations associated with segment s_i . GP_2 is fitted to the logarithm of the residual noise, $z_i = \log(r_i)$,

$$z(\mathbf{s}) \sim \text{GP}_2(m_2(\mathbf{s}), k_2(\mathbf{s}, \mathbf{s}')) \quad (1.5)$$

The mean μ_2 and covariance function Σ_2 for GP_2 are computed in a similar manner to GP_1 using standard GP equations (Equations 1.2 and 1.3). The most-likely heteroscedastic framework uses the mode of GP_2 to estimate the noise for any given segment:

$$r(\mathbf{s}) = \exp(\mu_2(\mathbf{s})) \quad (1.6)$$

As all distributions are Gaussian, the mode of this distribution is the mean. The heteroscedastic model incorporates uncertainty about the measurement variance into a third GP model GP_3 with the posterior distribution:

$$p(P^* | s^*, \mathcal{D}) \sim \mathcal{N}(\mu^*, \Sigma^*) \quad (1.7)$$

where

$$\mu^* = M + K^*(K + R)^{-1}(\mathbf{P} - M) \quad (1.8)$$

$$\Sigma^* = K^{**} + R^* - K^*(K + R)^{-1}K^{*T} \quad (1.9)$$

The notation defined for Equations 1.2 and 1.3 is augmented by:

$$R = \text{diag}(\mathbf{r}) \text{ with } \mathbf{r} = (r(s_1), \dots, r(s_n))^T$$

$$R^* = \text{diag}(\mathbf{r}^*) \text{ with } \mathbf{r}^* = (r(s_1^*), \dots, r(s_n^*))^T$$

The algorithm converges by setting $\text{GP}_1 \Leftarrow \text{GP}_3$ and iteratively updating the empirical noise. The posterior distribution in Equation 1.7 can be given to the planner or end user to make informed decisions based on power consumption estimates in different parts of the map.

In this framework, it is possible to use several covariance and mean functions. In our implementation, we use a Matern 5/2 kernel and linear mean function for both GP_1 and GP_2 .

Algorithm 1 Power Consumption Prediction

Input:

$$\mathcal{D} = \{(s_i, p_i)\}_{i=1}^n$$
$$\mathbf{s}^*$$

 $\triangleright$ Data Input $\triangleright$ Segments to calculate Posterior for**Output:** μ^*, Σ^* $\triangleright$ Posterior of required power consumption at segments $\mathbf{s}^*$ **Calculate GP₁**

1: $P(\mathbf{s}) \sim \text{GP}_1(m_1(\mathbf{s}), k_1(\mathbf{s}, \mathbf{s}'))$

2: $\mu_1(\mathbf{s}^*) = M^* + K^* K^{-1}(\mathbf{P} - M)$,

3: $\Sigma_1(\mathbf{s}^*) = K^{**} - K^* K^{-1} K^{*T}$.

4: **repeat****Calculate GP₂**

5: $r_i = \frac{1}{n_i} \sum_{j=1}^{n_i} (P_j - \mu_1(s_i))^2$

6: $\mathbf{z} = \log(\mathbf{r})$

7: $z(\mathbf{s}) \sim \text{GP}_2(m_2(\mathbf{s}), k_2(\mathbf{s}, \mathbf{s}'))$.

Calculate GP₃

8: $\mu^*(\mathbf{s}^*) = M^* + K^* (K + R)^{-1} (\mathbf{P} - M)$

9: $\Sigma^*(\mathbf{s}^*) = K^{**} + R^* - K^* (K + R)^{-1} K^{*T}$

10: **if** unconverged **then**

11: $\text{GP}_1 \leftarrow \text{GP}_3$

 $\triangleright$ So that $\mu_1(\mathbf{s}^*) = \mu^*(\mathbf{s}^*)$ and $\Sigma_1(\mathbf{s}^*) = \Sigma^*(\mathbf{s}^*)$ 12: **end if**13: **until** convergence

1.4 Experiments

To evaluate the accuracy of the power and energy consumption predictions, we collected data while manually driving a mobile robot in the centre of Oxford. The robot was equipped with current and voltage sensors (operating at 100Hz), and a Global Positioning System (GPS) receiver (operating at 1Hz).

1.4.1 Platform

The robotic platform (Figure 1.2) used in these experiments is a custom made platform developed within the EC-funded project EUROPA2.0 [eur, 2013]. The robot is designed to navigate autonomously in urban environments and provide assistance to pedestrians.



Figure 1.2: Europa Pedestrian Robot used for data collection.

1.4.2 Map Features

Path segment information is extracted from OpenStreetMaps and GoogleMaps. The complete list of features used in the predictive model is shown in Table 1.2. These features were chosen as they were publicly available and not collected by us, this allows our algorithm to operate anywhere with OSM data, not just where we happen to have collected Lidar or camera data. We include the number of surrounding amenities as a proxy for pedestrian traffic. This congestion would increase power consumption as the vehicle stops and starts to avoid people. By the same logic, we consider the presence of traffic lights and pedestrian crossing as they might also cause the robot to stop and start again to cross the street.

Our dataset includes five different routes (shown in Figure 1.3 and summarised in Table 1.1).

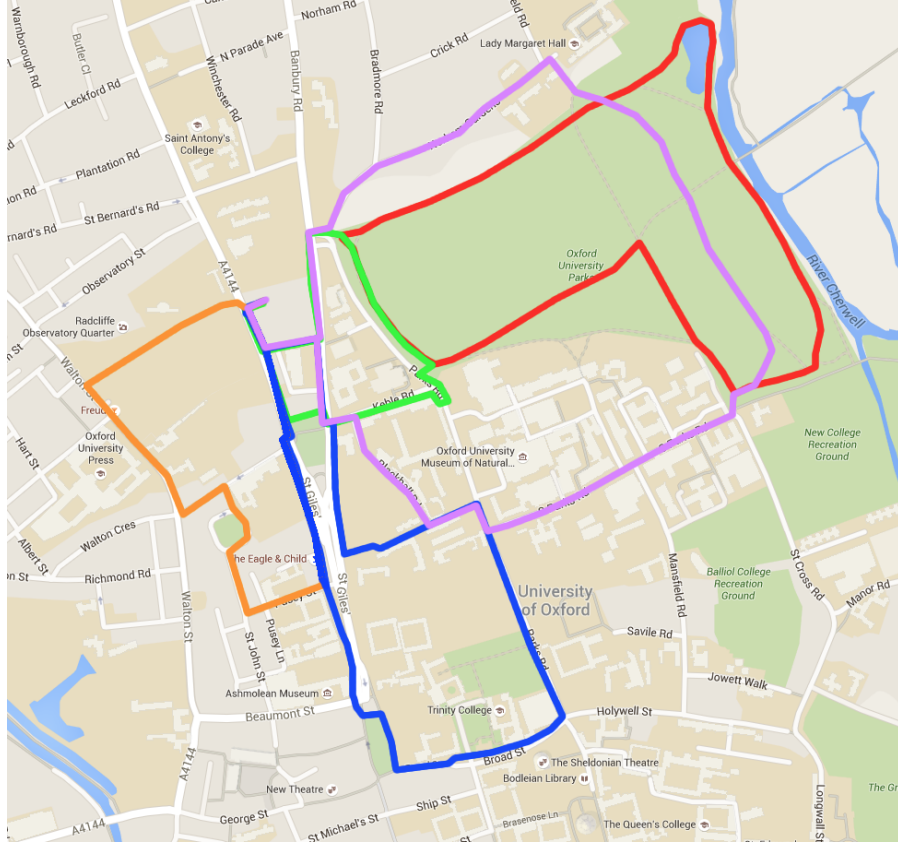


Figure 1.3: The five routes traversed by the robot and considered in the experiments. The segments comprising each route are not shown in this figure.

Route	Description	Length (km)
Blue	Passes through a pedestrian area with a high number of amenities	2.59
Orange	Quieter pedestrian route, low traffic	1.62
Purple	Long section along residential streets & gravel paths in park	2.30
Red	Open gravel paths through a park	2.89
Green	Residential streets, short park section and multiple crossings	1.51

Table 1.1: Summary of training routes

1.4.3 Data Collection

Our data collection process took place over two months and recorded over 30km of driving data in a range of different conditions. In total there were 1720 average power measurements on 820 unique segments, representing a broad spectrum of the type of paths in an urban environment would be expected to cover.

FEATURE NAME	DESCRIPTION	SOURCE
SURFACE (SRF)	terrain type (asphalt, gravel, ..)	OSM
PATH TYPE (PT)	path type (sidewalk, pedestrian area, ..)	OSM
INTERSECTION (INTN)	number of intersections along the path segment	OSM
CROSSING FLAG (CF)	presence of traffic lights or crossing facilities	OSM
AMENITIES (AMN)	surrounding amenities within 100m (museums, pubs, shops, ..)	OSM
ELEVATION (EL)	change of elevation along the path segment	Google Maps

Table 1.2: Description of the features used to characterise the path segments and build the predictive model.

1.4.4 Sensor Data Post-Processing

Recorded sensor data are notoriously noisy, which arises from a number of different sources. These errors can be either systematic (a change in the environmental conditions, such as temperature) or due to random effects (see [Elnahrawy and Nath, 2003]). Several techniques have been proposed in the literature to post-process sensor data and obtain reliable measurements (see [Basu and Meckesheimer, 2007, Weiss et al., 2012]). The current and voltages are measured every 100Hz, but the raw measurements are very noisy. Following [Weiss et al., 2012], we apply a median filter followed by a mean filter to the current and voltage sensor readings to remove outliers and high-frequency noise from the data. For both the median and mean filters, we empirically chose a window of five samples, which was able to remove high frequency sensor noise but allow the underlying signal to be accurately measured. The processed current and voltage measurements were multiplied, and this processed power signal was averaged to give power consumption measurements p_i for each segment s_i . If a segment was traversed multiple times, multiple power measurements were recorded, so that the model could learn variations in power readings.

Noisy GPS measurements are matched to map segments using the Hidden Markov Model (HMM) based approach proposed in [Newson and Krumm, 2009]. The power measurements are associated with the respective segment using the timestamps of the GPS readings and sensors. The power consumption of the entire route is obtained by averaging the power consumption of each segment along the route.

1.5 Results

To evaluate our system, we perform three different experiments. We first analyse the performance of the heteroscedastic regression which predicts power consumption on unseen paths. Later, we introduce a simple benchmark model and compare the predictions provided by such an approach with those provided by our model. This highlights the pertinence of map features for predicting power consumption. Lastly,

we evaluate the relevance of those individual features within our model.

1.5.1 Energy Consumption Predictions via Heteroscedastic Regression

The purpose of this experiment is to evaluate the accuracy of the power and energy consumption predictions provided by the heteroscedastic regression model on routes that are not observed in training. To do so we perform leave-one-out cross-validation. We do this cross-validation on a route-by-route basis rather than a segment-by-segment basis as this evaluation is closer to the use-case of this algorithm, making predictions on an entirely unseen route, not just portions of it. We train our model with data from every traversal of every route, excluding the route under test, and we compare our predictions against the measured power from all traversals of this route. We measure our performance using the relative error between the predictions and measurements. The results are shown in Table 1.3. Our method provides

Route	P Measured [W]	P Predicted [W]	μ error [%]	σ error [%]
Blue	561.97 ± 4.62	557.90 ± 4.32	0.7	6.5
Orange	580.92 ± 10.09	556.05 ± 3.52	4.2	65
Purple	541.10 ± 19.59	549.48 ± 2.85	1.5	85.4
Red	565.46 ± 2.21	539.64 ± 2.09	4.6	5.4
Green	531.42 ± 23.18	560.37 ± 3.44	5.4	85.1
Average	-	-	3.31	49.5

Table 1.3: Comparison of Measured and Predicted Power Consumption for different routes, using the heteroscedastic GP approach.

estimates with an average relative error of 3.31% on the average power estimate and an average relative error of 49.5% on the predicted variance.

As well as performing well on each route, the model also obtains accurate power consumption estimates on a segment by segment basis. Figure 1.4 illustrates the behaviour of our regression model on the five routes individually, showing the power consumption predictions for each segment in each route. We can see that the high variance in the data is well captured by the model and the majority of the power measurements lie within two standard deviations. In some cases, there is bias in the mean prediction that we observe which is due to the evaluation framework chosen. By performing leave-one-out cross-validation, the training set is not always representative of the route travelled. For instance, the red route consists of many gravel segments which are removed by leave one out cross validation, leading to the under-estimate of required power.

Accurate energy estimates for each segment can be calculated by combining our model with a speed profile. We implement a basic speed profile by assuming that the robot always travels at a constant speed, v_{avg} , corresponding to the average speed recorded on the training routes. The energy required to traverse segment i of length l_i is

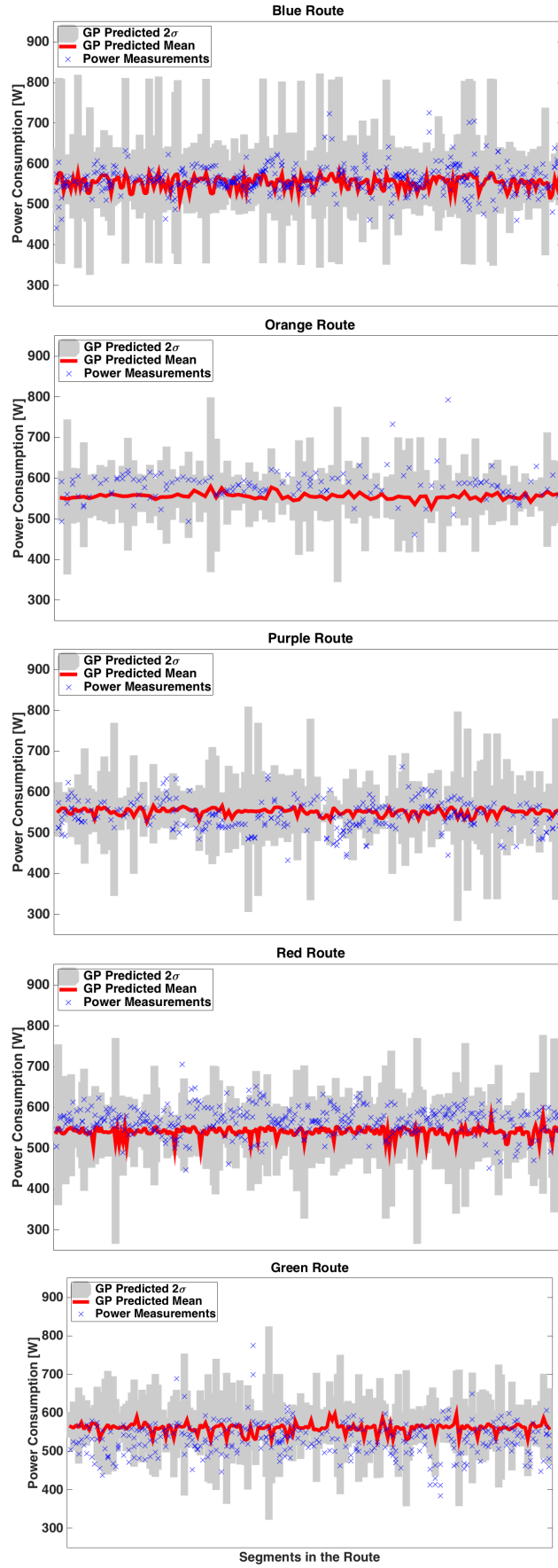


Figure 1.4: Power consumption predictions against actual power measurements for each segment of each route. The continuous red line indicates the mean predictions provided by the model, the grey shade indicates the predicted 2 standard deviations. Actual power measurements are indicated by blue crosses.

$$\hat{E}_i = P_i \times \frac{l_i}{v_{\text{avg}}} \quad (1.10)$$

The results are shown in Table 1.4 which compares the energy consumption measurements E and the energy consumption estimates $\hat{E}$, calculated by summing all segments in the route. The relative error on the average energy consumption is 6.6%. This suggests that, even with a naive speed profile, it is possible to obtain accurate estimates of the average energy consumption. More advanced speed profiles can only improve this accuracy.

Route	E [Wh]	$\hat{E}$ [Wh]	μ error [%]
Blue	394.07 ± 7.61	393.93 ± 2.82	0.03
Orange	294.94 ± 25.35	366.4 ± 2.17	24.2
Purple	354.99 ± 54.36	360.90 ± 1.74	1.66
Red	628.78 ± 125.93	627.51 ± 2.07	0.2
Green	260.29 ± 29.97	278.72 ± 1.70	7
Average	-	-	6.64

Table 1.4: Comparison of Measured and Predicted Energy Consumption for different routes, using the heteroscedastic GP approach.

1.5.2 The Benchmark

In order to demonstrate that the map features contain information relevant for predicting power consumption, we compare the proposed framework with a benchmark approach. The benchmark approach assumes there are no significant differences amongst the power requirements of different path segments and estimates the power consumption of unknown routes without taking the segments' characteristics into consideration. The benchmark approach uses the mean and variance of the traversed routes as the mean and variance prediction for the unseen route, making the assumption that all segments are the same. We evaluate the behaviour of this model in a leave-one-out cross-validation framework. The obtained results are shown in Table 1.5. When compared to Table 1.3, Table 1.5 demonstrates the difference in performance between the benchmark approach and the proposed framework.

Route	P Measured [W]	P Predicted [W]	μ error [%]	σ error [%]
Blue	561.97 ± 4.62	550.14 ± 24.82	2.10	436.7
Orange	580.92 ± 10.09	548.11 ± 20.88	5.6	106.9
Purple	541.10 ± 19.59	555.87 ± 22.80	2.7	16.4
Red	565.46 ± 2.21	549.32 ± 24.38	2.9	1003.2
Green	531.42 ± 23.18	560.13 ± 17.76	5.4	23.38
Average	-	-	3.75	317

Table 1.5: Comparison of Measured and Predicted Power Consumption for different routes, using the benchmark.

The relative error on the estimates of the mean power using the benchmark method is 3.75%, while the average relative error on the variance prediction is 317%. Our proposed framework shows an improvement of 11% in the accuracy of the prediction of the mean, and an improvement of 84.3% in the accuracy of the

prediction of the variance. We attribute this improvement to the fact that the benchmark approach does not take into account the specific characteristics of the segments, but considers all routes and segments equivalent. By incorporating knowledge about the features of each path segment, our proposed methodology is able to model the power consumption and its variance with higher accuracy.

1.5.3 Map Feature Ranking

Having shown the information content in the map features, the last experiment aims to evaluate the predictive power of each feature in the GP model. In a leave-one-out cross validation framework, we use the heteroscedastic regression approach to estimate the power consumption using only a subset of the features. We compare the predictions made with all the available features (as a baseline) with predictions made after removing individual features to judge their predictive power. The decrease in accuracy is expressed as the relative error between the two predictions. Figure 1.5 shows the result of this analysis. Excluding any of the features has a negative impact on the accuracy of the power consumption predictions; the most discriminant features in our framework are the number of surrounding amenities and the path type. This corroborates our experiences whilst testing, two major reasons for stopping and starting were to avoid pedestrians and to manoeuvre through narrow, awkward streets.

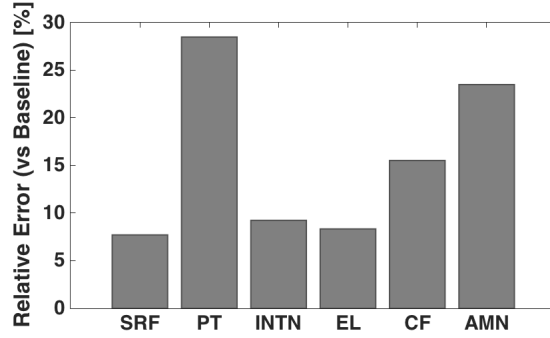


Figure 1.5: Comparison between the relative error on the variance prediction, when different subsets of features are available. On the y-axis, the relative error is computed with respect to the one given when all features are considered (cfr. Table 1.3). The labels on the x-axis indicate the removed feature (cfr. Table 1.2 for full explanation).

1.6 Conclusions

This work addresses the problem of estimating energy consumption of a mobile robot moving in an urban environment. Previous studies investigate how energy consumption is influenced by the kinematics of the robot, its processing and perception. However, using high-level features extracted from publicly available maps, we provide a probabilistic model that accurately predicts energy consumption. As our feature selection is open source, we can extrapolate our predictions to routes that have not been seen previously. Furthermore, as the model does not require information about the internal configuration of the robot, it is applicable to a wide range of platforms. The proposed framework is implemented with Heteroscedastic Gaussian Processes to better represent the variance in the data, which is difficult to model directly. The distinct advantage of our model is our ability to predict measurement variance. Our estimates outperform benchmarks by 84.3%. Such accurate estimates are particularly useful for any planning system that needs to make confident decisions while dealing with limited robot power supplies.

Further investigation could focus on enriching the feature set by extracting information from other sources which might be better predictors of robot efficiency. For example, using Google Streetview data or disabled access data could reveal the state and upkeep of pavements that the robot traverses.

Chapter 2

Learning Terrain Drivability Online using Laser for Self-Supervision

2.1 Introduction

Chapter 1 showed that high level drivability could be estimated by using publicly available maps. However, as explained in the introduction, this method would not be sufficient in cases when map annotations are not available.

In 2016 the Oxford Robotics Institute started collecting data in the Wychwood Forest, part of the Cornbury Park estate. As the estate is private property, OSM annotations are very poor (Figure 1), both in terms of the different potential routes and the inherent features of those routes and the environment. Added to this, the platform had to be switched from the urban pedestrian Europa Platform to a 4x4 vehicle which could cope with the off-road conditions.

In short, very little information was available about the performance of the platform and the environment ahead of time. We would still like to be able to operate the robot without this prior knowledge of platform and environment, which motivated us to ask the larger research question - "Is it possible to operate a robot in an environment where prior information about where to drive is unavailable?".

This Chapter explores two fundamentally different approaches to answering this problem. Firstly, we benchmark models trained ahead of time on similar datasets to determine whether they are able to generalise where the test set diverges from the training set.

The advantage of these models is that there are no resource constraints for training the model ahead of time, they can be given as much training data as has been collected elsewhere, however there are no guarantees about the performance in a novel environment. A model trained in a desert might fail in a

forest, a model trained in a forest might fail in the snow. It is impossible to know whether a ‘static’ model will work without bringing the robot into the new environment and testing it, and at this stage it is too late to retrain or change it if it fails. These problems stem from breaking a key assumption of machine learning, that the distribution of data in the training and test set must be the same.

To bypass this problem we investigate algorithms which utilise the data collected by the robot at runtime as it moves through the environment. As this training data is collected by the same platform and sensors, in a similar environment under similar conditions, there is a stronger guarantee that the distributions are the same.

One problem with using an online algorithm is that training data needs to be labelled. Previous work in segmentation is often labelled by hand, which would be impractical with a robot in the field. We propose a method which uses a push-broom laser to label drivability within a scene. We accomplish this by using the vertical variance of laser points in a pointcloud (See Figure 2.3). Using this metric as a proxy for drivability captures the idea that flat ground is drivable, whereas rough or sloping ground is not. This method is not platform specific, doesn’t rely on hand tuning parameters, and similarly to Chapter 1, allows us to collect and leverage real-world data automatically.

We investigate two algorithms which use this self-supervised data. Firstly, following [Dahlkamp et al., 2006] we keep training and using models with the most recent data. However, this seems computationally inefficient, so we also propose a method which uses image similarity to trigger whether to change or update the model, in the same Experience Based spirit as [Churchill and Newman, 2013] and [Hawke et al., 2016].

We seek to answer the question: “If a robot is operating in a new environment using a drivability prediction model, what is the most pertinent data for training that model? Is it possible to leverage the data the robot records in an online fashion?”

To this end we use data collected in Cornbury Park and two other locations in Oxford to compare the performance of these different potential models.

2.2 Related Work

Learning Off-Road drivability is not a new problem in robotics. Much of the early research to show promise centred around the 2004 and 2005 DARPA Grand Challenges which saw teams develop autonomous cars to perform in previously unseen off-road environments. This included teams from Stanford, CMU and Cornell [Thrun et al., 2006, Kelly, 2006, Bibuli et al., 2007, Shneier et al., 2009]. One resulting method uses laser to label drivable paths close to the vehicle then trains a model to propagate these labels further away using monocular cameras [Dahlkamp et al., 2006]. Other “Near-to-far” algo-

rithms include [Hadsell et al., 2009] who extend colour histograms to large image patches, [Bajracharya et al., 2008] who incorporate a polar-perspective map to aid far-field planning, and [Happold et al., 2006] who combine unsupervised color to geometry models with supervised geometry to traversability models to learn image to traversability models. [Suger et al., 2016] use a lidar model which is adapted to potential environments (urban, off-road, forest etc.) These models are often applied naively, they have no notion of when retraining a new model or augmenting an existing model is necessary. This can lead to high computational loads as models are continuously tweaked and updated.

Methods which incorporate Deep Learning for off-road driving and path recognition include end-to-end training for obstacle avoidance [LeCun et al., 2006] and rough direction finding for a drone in a wooded path [Guisti et al., 2016]. [Barnes et al., 2017] automatically collect data which they use to train a path-proposal network, however this approach assumes the ground is planar in front of the vehicle, an assumption not necessarily true in off-road environments when roads can be more undulating. Deep Learning approaches are state of the art in many computer vision tasks such as object detection [Han et al., 2018] and semantic segmentation [Garcia-garcia et al., 2018]. However, the predictive power of these methods are offset by their computational requirements, particularly at training time. As a result DNNs are rarely deployed dynamically, updating and retraining at runtime, although attempts have been made to adapt lower levels at runtime [De Brabandere et al., 2016].

Using different sensor modalities is a common aid for classifying drivability, including Camera-LIDAR fusion [Häselich et al., 2011, Laible et al., 2013], proprioception [Brooks and Iagnemma, 2012], vibration [Otsu et al., 2016] and sound [Libby and Stentz, 2012]. However, the problem with these methods is often that a robot needs to physically interact with terrain before it's classified as inappropriate, which can be somewhat dangerous. [Berczi et al., 2015, Berczi and Barfoot, 2016] do terrain assessment in a Visual Teach and Repeat (VT&R) framework, utilising the fact that a route is initially driven and deemed safe by a human. [Ho et al., 2013] use Gaussian Processes to extrapolate elevation maps in planetary rovers. [Ross et al., 2015] estimate the ground-height under vegetation in order to determine traversability.

Novel terrain detection is a related subset of off-road driving, research includes using Support Vector Machines [Brooks, 2009], attaching a one-vs-all classifier to a conv-net to overcome the closed-set limitation of convolutional neural network architectures [Sünderhauf et al., 2015], using auto-encoder reconstruction loss [Richter and Roy, 2017].

2.2.1 Motivation

Section 2.2 outlines a few of the different approaches for predicting traversability in a mobile robot's environment. Broadly these can be split into those that learn and adapt online and those trained beforehand

and are static at runtime. The advantages of these models is that it is the computationally heavy training can be performed on more powerful computers ahead of time, training can take longer, and more advanced algorithms be trained.

The underlying problem with an unchanging model is that it relies on the assumption that the environment in which it has been trained is statistically comparable to the runtime environment. If this assumption is untrue, and if it doesn't perform well at runtime, there is no scope to adapt it.

The other family of solutions are those that learn and adapt online. However, they have no notion whether retraining is necessary or not, they naively continue to retrain models, which can be computationally expensive. This is a problem on energy constrained robots, where continually running computationally intensive training can reduce the effective range of the robot.

Our proposed algorithm attempts to sit between these two approaches, offering the flexibility of solutions which adapt to new environments, but in a way that is computationally efficient, only training new models when current models are no longer suitable.

Learning traversability could happen in laser point clouds or in images. We choose images for our pipeline, because processing pointclouds is still an expensive operation, 3D lidars are still more expensive than cameras, and the resolution of pointclouds decreases further away from the robot, making lidar less informative than cameras at long ranges.

The assumption is that the resulting model would be used within a full drivability pipeline, perhaps informing a separate planner of the cost of traversing different routes, or even as a "sense-check" of more complex, static models, handing over to an operator if the two sets of predictions significantly diverged.

2.3 Framework Description

Section 2.3.1 describes the methods used to generate real world training data automatically to avoid the need to hand-label images. This allows the robot to automatically generate proxy-drivability data as it traverses the environment, in a similar manner to Chapter 1. Section 2.3.2 explains the Random Forest model and features selected to predict drivability from images. Section 2.3.3 introduces an online Global Image Descriptor, a constituent part of the overall algorithm described in Section 2.3.4.

2.3.1 Using Laser Variance as a proxy for Drivability

The method proposed seeks to leverage the most recent data collected by the robot in order to decide drivability. There are two main difficulties to overcome. Firstly how to collect negative training examples. The robot experiences many examples of drivable paths and terrain, but experiencing undrivable paths might risk damaging it.

The second issue is to remove the need for human labelling. Traditional methods using human anno-

tation of data are infeasible for online learning in the field. Alternatively we need to rely on other sensor modalities to label drivability as outlined in Section 2.2. However there is no guarantee that their models are interpretable, applicable or sensible in different test domains.

We propose using vertical variance of laser points to generate self-supervised training data within an image-to-drivability pipeline. These laser points are generated with a 2D laser in a push broom configuration, mounted perpendicularly to the direction of motion. Successive scans are composed together using Visual Odometry (VO) and projected back into a camera frame from the start of the swathe (See Figure 2.3). This has the same flavour as [Dahlkamp et al., 2006] who used a forward facing laser to label a scene, rather than a push broom laser. 2D lasers are cheaper than full 3D lasers and are able to capture higher fidelity pointclouds by concatenating multiple scans. Hereafter ‘swathe’ shall be used to refer to a collection of laser points and the associated image into which they are projected.

The training data within swathe S consists of image features $X_{1...n_S}$ (See Section 2.3.2) and vertical variance $Y_{1...n_S}$ for all n_S laser points that are reprojected into the swathe image. Vertical variance is defined as:

$$Y_i = \text{var}(h_1...h_j) \quad (2.1)$$

where $h_{1...j}$ are the vertical heights of all laser points within a euclidean distance $d_{\text{laser window}}$, as shown in Figure 2.1. If there are no other laser points within this window, variance cannot be calculated and the point is not included in the training data. This spatial windowing trades off between overall smoothness and accuracy. Figure 2.1 shows this calculation for a flat road section and for a tree. This neighbourhood windowing is only along a single scan line to increase the speed of the algorithm, reducing a 2-D search for nearby points to 1-D. No qualitative loss of performance was observed for this speedup, particularly given the KNN smoothing described below. To aid the Random Forests and to make figures clearer, vertical variances beyond a threshold $\theta_{\text{threshold}}$ are capped $Y_i = \max(Y_i, \theta_{\text{threshold}})$. If the threshold is not incorporated, the limited model capacity focuses on learning to predict how undrivable an area is. However we are less concerned with predicting drivability in an area that we are unlikely to ever consider traversing, so we incorporate this threshold so that model capacity is focused towards areas which are more marginally drivable, and would be of greater interest to a planning algorithm in the pipeline.

Hereafter vertical variance will be referred to as drivability for more convenient nomenclature.

To reduce the noise in our data, a K-Nearest Neighbour filter was applied in image space (See Figure 2.2). This is to remove areas classified as drivable where spurious laser points happen to have low vertical variance but are surrounded by other laser points with high variance.

An additional benefit of using this method is its simplicity. Variance is unparameterised and captures

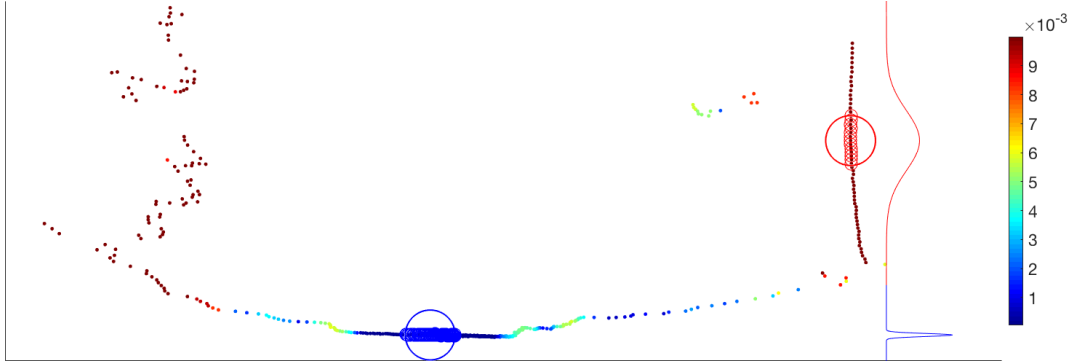


Figure 2.1: Diagram showing how vertical variance is calculated in a single 2D laser sweep. The swathe consists of a bush (left), a road surface (centre) and a tree (right). Laser point colours denote the calculated vertical variance. Blue and Red circles indicate how points within distance $d_{\text{laser window}}$ are used to calculate variance, and the resulting variances are shown on the right hand side (not to scale). The blue points have a low vertical variance, whereas the red points have a high vertical variance. Also note the edges of the roads are assigned an intermediate drivability.

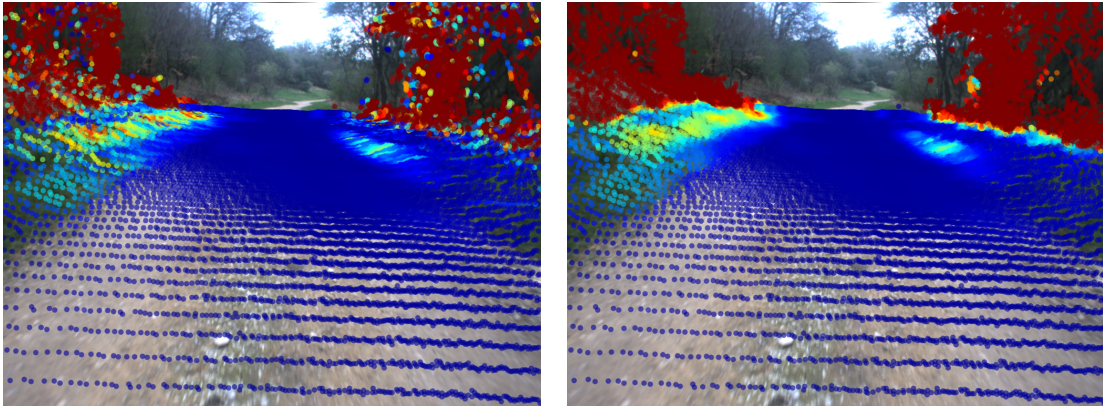


Figure 2.2: The effect of KNN Smoothing on Swathe Data. Blue indicates high drivability, red indicates low drivability. Left: Without Smoothing. Right: With KNN Smoothing. Note the removal of spurious ‘drivable’ points in the bushes on either side.

two major aspects of drivable terrain. It penalises smooth, steep areas (such as sloped banks), as well as rough but flat areas (such as long grass). Having one statistic to encapsulate the different factors which effect traversability removes the need for an operator to tune parameters with a platform in the field. Figure 2.1 shows that vertical variance is able to capture these areas of higher gradient and surface roughness.

One by-product of these choices is that terrain assessment, traditionally a classification problem, is now formulated as a regression problem. Rather than defining drivability as binary (drivable or undrivable), drivability now exists on a continuum. This corroborates human experience of driving, for instance, long grass is drivable or undrivable only in the context of what other terrain exists in the scene. It is a more preferable terrain than large bushes, but not as preferable as driving on tarmac. As such we recast our prediction problem as regression rather than classification.

By using this method, the algorithm is able to leverage the most recent and relevant training data without relying on human labelling or handcrafted parameters. Rather than relying on prior models of drivability which may or may not be either interpretable or applicable to the current environment, an operator need only ask whether the vertical noise assumption is a good proxy for drivability. There will be

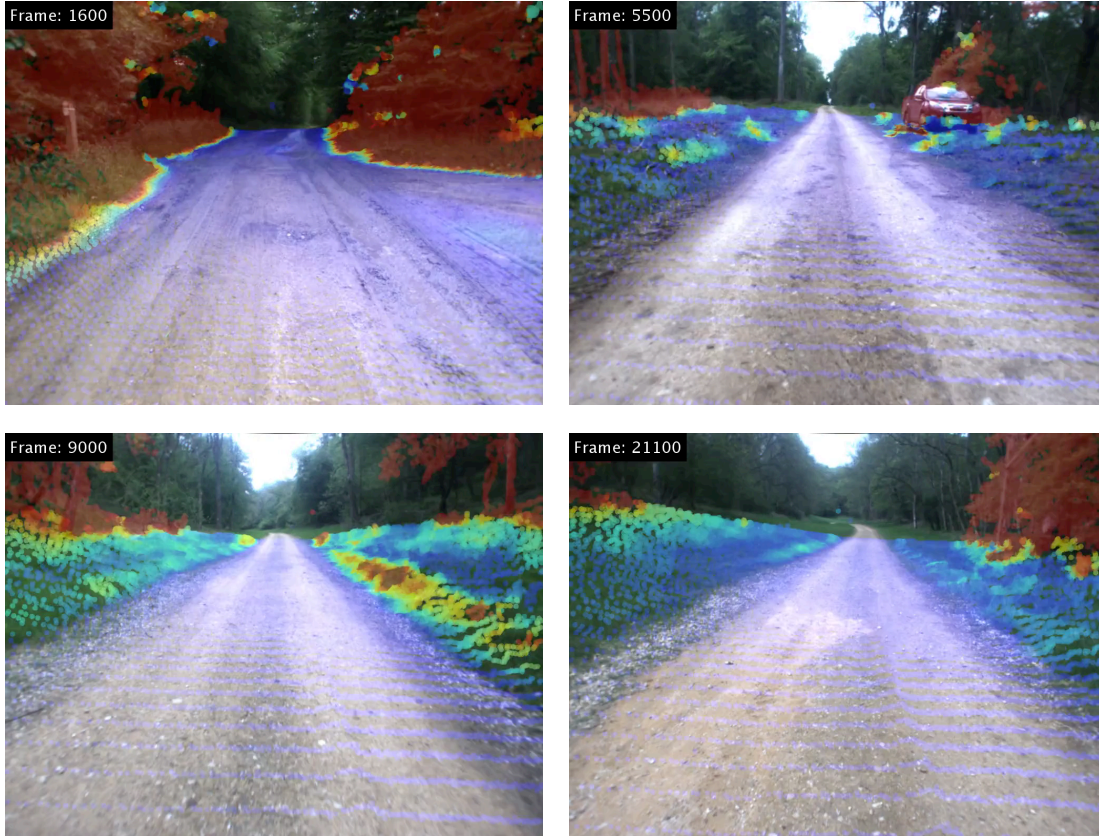


Figure 2.3: Examples of vertical variance used as a proxy for drivability. Blue indicates high drivability, red indicates low drivability. Successive frames are concatenated with VO and back-projected into an image taken at the start of the swathe. To show the underlying scene, the laser points are partially transparent.

environments for which this is not the case, for instance driving across an icy lake where vertical noise is not a good indicator of ice thickness and safety, or driving in urban environment where flat pavement and flat road surface look equally viable. However these situations were not encountered in the dataset and are outside the scope of this work.

2.3.2 Image to Drivability model

Given this source of online training data, Random Forests are used to predict drivability in new image frames. Random Forests are commonly used in image classification and regression tasks (For example [Paz et al., 2016, Scharwachter and Franke, 2015]). They are fast to train, fast at run-time and do not require manual architecture crafting or hyper-parameter tuning, the major design decision is the number of trees in each forest which depends on the required speed and accuracy of the algorithm.

Recently Deep Neural Networks have made a resurgence in the Computer Vision field. They are not implemented in this algorithm for a number of reasons. Firstly because they are expensive to train both in computational resources and data required. This precludes them from use in an online, iterative algorithm. However, if methods became available to train them more cheaply, nothing would prevent the algorithm presented in 2.3.4 from making use of them if they were found to be more effective than the current RFs.

The downside of using Random Forests over Deep Networks is that image features have to be chosen

a-priori, rather than learned in an end-to-end manner. This thesis uses 28 feature channels (Shown in Figure 2.4) There are 7 different inputs at 4 different scales. The inputs are the original RGB colour space, an illumination-invariant colour space [Maddern et al., 2014] and the LAB colour space. Pyramid Scaling [Adelson et al., 1984] is used to generate 4 different input scales to capture contextual information around each pixel.

This work started with RGB features although they struggled with shadows and lighting changes. As a result the (L)AB and INV channels were included in an attempt to add invariance to lighting changes.

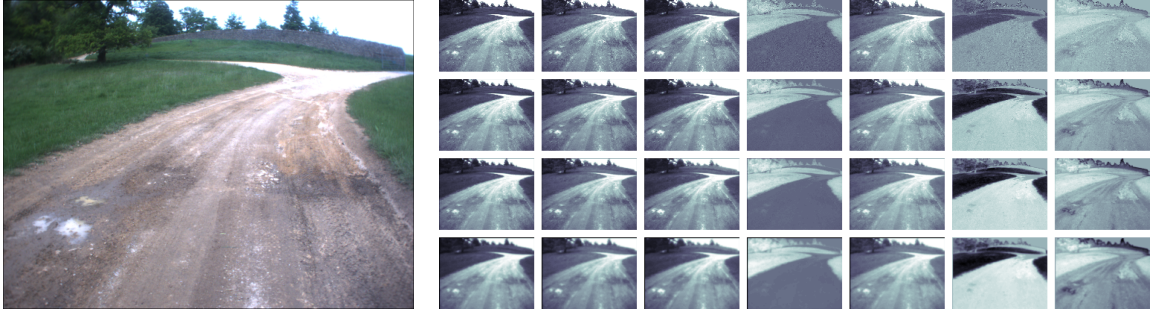


Figure 2.4: Left: Input Image. Right: Extracted Feature Channels. From left to right, RGB, Invariant [Maddern et al., 2014], LAB. From top to bottom: High resolution to low resolution image scales [Adelson et al., 1984].

The image features selected are by no means optimised for this project. The aim of this project was to investigate whether an Experience Based approach was suitable in off-road environments, not to optimise components of underlying models such as image features. For this reason the same image features are kept constant and used in the benchmark methods.

One property of using a Random Forest model is that the importance of each feature can be determined from training with ‘out of bag’ sampling [Altmann et al., 2010]. Figure 2.5 shows the importance of each of the 28 Features when a Random Forest is trained on data sampled from all datasets.

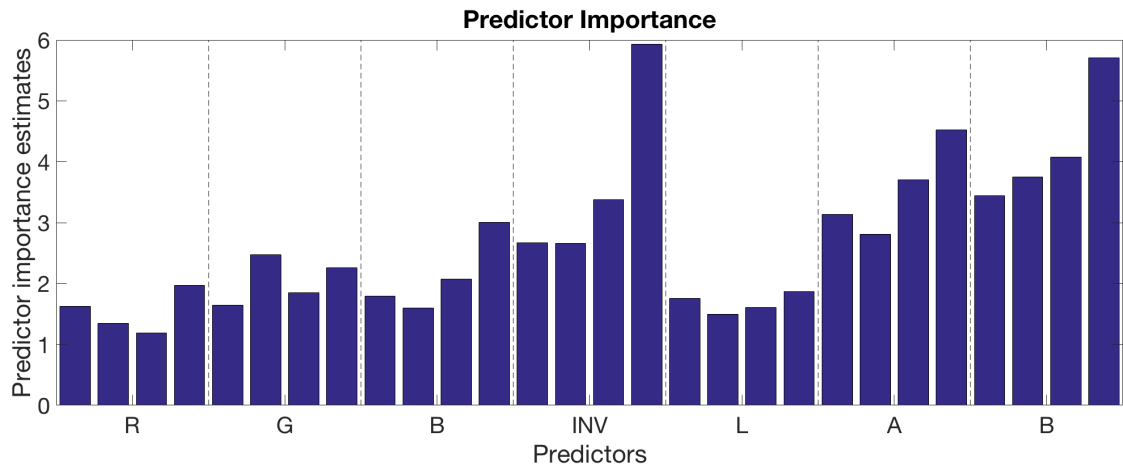


Figure 2.5: Image Feature Importance, calculated with out-of-bag permutation¹. Each channel has four bars, from high resolution to low resolution (left to right). Most useful features were INV, A & B at lower resolutions. Least useful features were Luminance & RGB. Generally speaking, lower resolution, higher context features were more useful than high resolution individual pixel values.

Having decided a set of image features, a Random Forest model can be associated with each swathe S , to make a prediction

$$\hat{Y}_i = \text{RF}(X_i, \theta_S) \quad (2.2)$$

on image features X_i at a sample point i , where θ_S are the Random Forest parameters optimised for the training data in swathe S .

2.3.3 Online Global Image Descriptors

Visual Place Recognition is a well-established field in Computer Vision [Lowry et al., 2016] that aims to predict whether two images are of the same place. This work has a similar application, however, rather than predicting whether two images are of the same place such as in [Cummins and Newman, 2008, Cummins and Newman, 2011, Arandjelovic et al., 2017], the aim is to predict whether one swathe’s model accuracy predicts drivability in a different swathe. Following [Mactavish et al., 2017], this paper uses cosine distance between Bag-of-Word (BoW) descriptors as a metric of image similarity. Traditionally the dictionary of words used in these descriptors is given a-priori. However, when operating in a new environment there is no guarantee that the chosen dictionary is still representative, so we propose a method which learns the dictionary online.

Every x m along the robot’s path, a new image is shown to the algorithm. This spatial sampling helps to avoid oversampling when the robot is stationary and provides a representative sample of the environment. The same image features used in Section 2.3.2 are extracted from the image. Reservoir Sampling [Vitter, 1985] is used to subsample a smaller set of features which is representative of the entire route.

The number of words in the vocabulary and the size of the reservoir are kept constant so that the algorithm speed stays constant and doesn’t slow down as the robot goes further and sees more images.

The vocabulary centres are updated by assigning all points in the reservoir to their respective centres and recalculating the mean position of the centre. This is equivalent to one step with Expectation-Maximization in K-means Clustering. To generate the cluster centres in the first iteration, all iterations of K-Means are used.

Alternative methods which were considered included adding new words to the vocabulary if they exceeded a distance threshold from existing words, however this parameter would require too much fine tuning to trade-off between having too many words (for computation) and too few (for performance).

To sense-check this method, we define a metric d as the average distance between each pixel and the

¹Out-of-Bag, Predictor Importance Estimates by Permutation:
(<https://uk.mathworks.com/help/stats/regressionbaggedensemble.oobpermutedpredictorimportance.html>)

nearest word in the vocabulary -

$$d = \frac{1}{P} \sum_{p=1}^P \operatorname{argmin}_{v \in V} (\|\mathbf{f}_p - \mathbf{f}_v\|) \quad (2.3)$$

where $\mathbf{f}_p$ and $\mathbf{f}_v$ are the feature vectors for the P pixels and V vocabulary words.

Figure 2.6 uses this metric to compare this proposed vocabulary generator with two static vocabularies as benchmarks. The first ‘Global vocab’ is the Vocabulary generated by running K-means clustering over image features sampled from every log. ‘Log Vocab’ is the vocab generated by running K-means over the single log ahead of time. We see that average distance between the image and the vocabulary is lower when the vocabulary is able to adapt online, even when the baseline is trained on the log ahead of time.

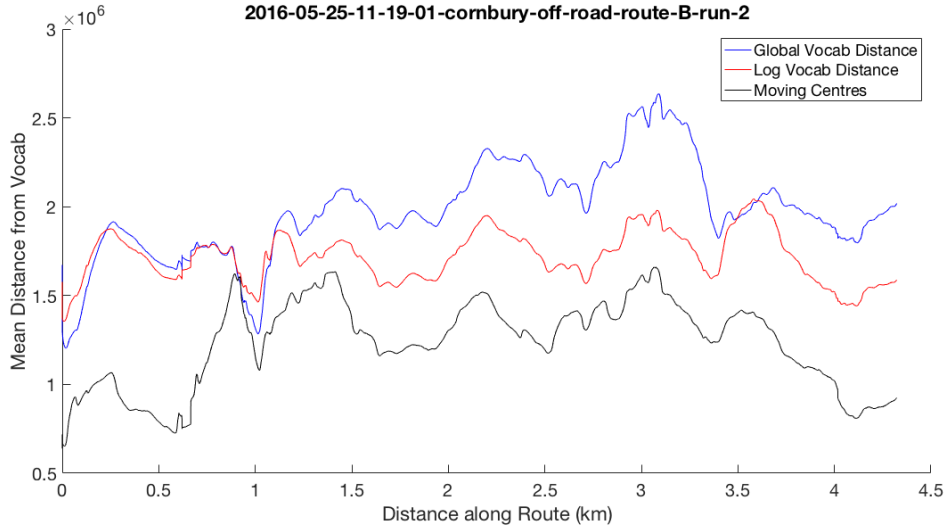


Figure 2.6: The vocabulary distance defined in Equation 2.3 for the proposed Moving Centres Method, a ‘Global’ vocabulary trained on all logs and a ‘Log’ Vocabulary trained solely on this log. The Moving Centres vocabulary is consistently much closer to the current frame.

One interesting side effect of using the metric in Equation 2.3 is that it correlates with unusual or unexpected parts of the image. One possible application is to use this for novelty detection or introspection. Although this application is not explored in this work, Figure 2.7 shows different examples of detecting unusual objects within scenes using this metric.

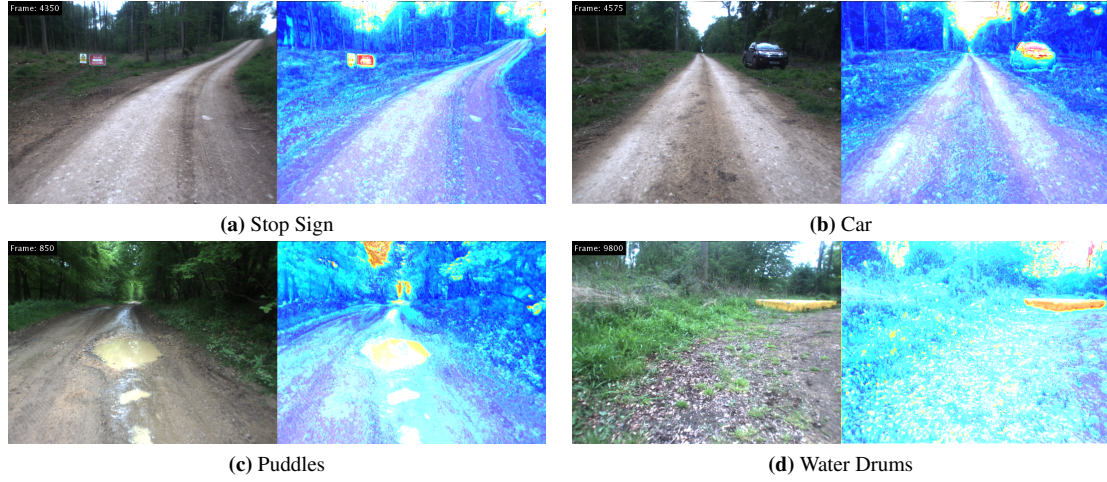


Figure 2.7: Examples of objects which could be classified as novel using distance from trained vocabulary metric.

2.3.4 Experience Based Off-Road Algorithm

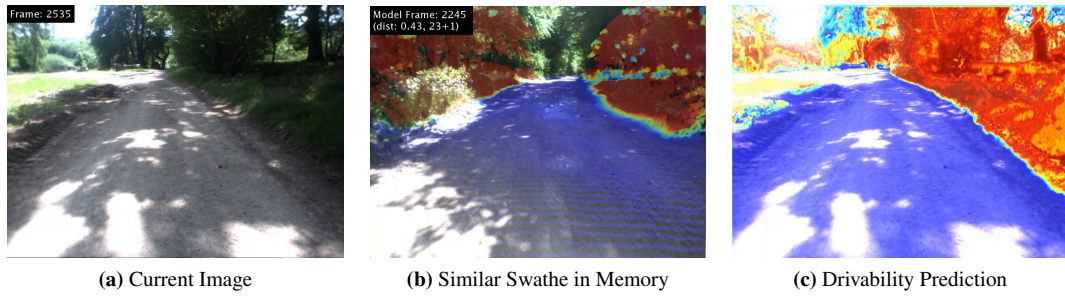


Figure 2.8: Using the Experience Based Off-Road algorithm, a suitable swathe is found for the current image and a prediction made with the associated model.

Sections 2.3.1, 2.3.2 and 2.3.3 introduce all the key components for the proposed system which is described here.

Experience Based Frameworks [Churchill and Newman, 2013, Hawke et al., 2016] provide a principled way for model size to scale with the complexity of a problem. Rather than train general models to perform in all environments, specific models are trained to perform well in specific conditions and environments. Larger and more complicated environments lead to a larger algorithmic footprint and a greater number of trained models, simple environments lead to a smaller footprint and fewer models. This paper proposes ‘Experience Based Off-Road’ (EBOR), a principled method of learning image-to-drivability models in off-road environments. As far as possible this happens with minimal prior information.

The way that the algorithm is initialised is by an operator manually driving a swathe to start. The robot trains a model on the data in this swathe and is able to use this model to traverse subsequent paths. If it encounters a path dissimilar to anything previously observed (The STOP condition in Figure 2.10b), it passes control back to the human operator to traverse this difficult path, trains a new model on this data and continues.

The reason for using this framework is to reduce the computational load of training and retraining new

models. Previous work in the area has no notion of when retraining is necessary, this framework retrains when a new image exceeds a distance threshold from previously trained models.

The framework consists of 3 different algorithms running in parallel. They are:

1. **Core loop:** The selected model is used to make predictions on the current image (Figure 2.9). No training takes place within this core loop as it needs to run fast.
2. **Swathe Retraining:** This ancillary Algorithm is the key novel insight developed by in this work and determines when new models need to be retrained. It is shown in Figure 2.10a and each operation outlined in full below.
3. **Vocab Retraining:** Using the method outlined in Section 2.3.3, this ancillary task updates the Vocabulary used by the robot for image similarity (Algorithm shown in Figure 2.10b). It also updates the image descriptors for each model in memory so that they can be compared to the current image.

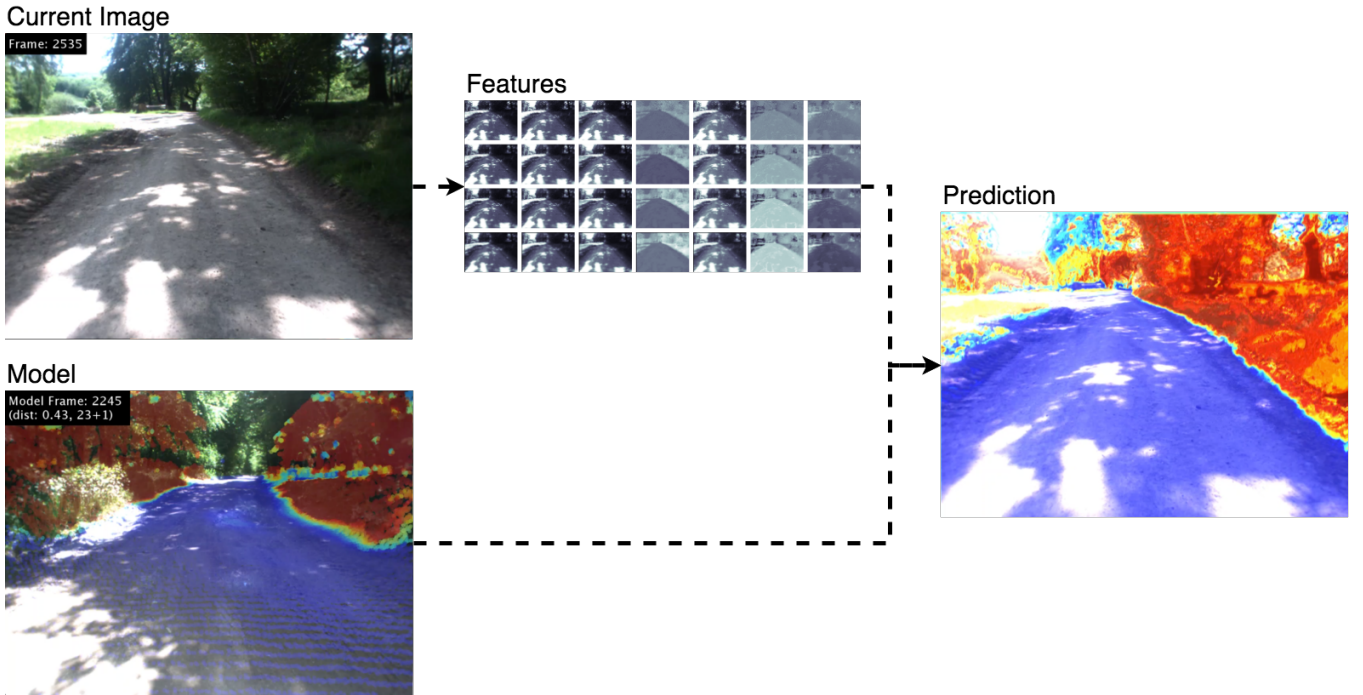


Figure 2.9: Core Loop. Features are extracted from the current image. A Random Forest Model, pretrained on an earlier swathe is run on these features to generate the drivability Prediction.

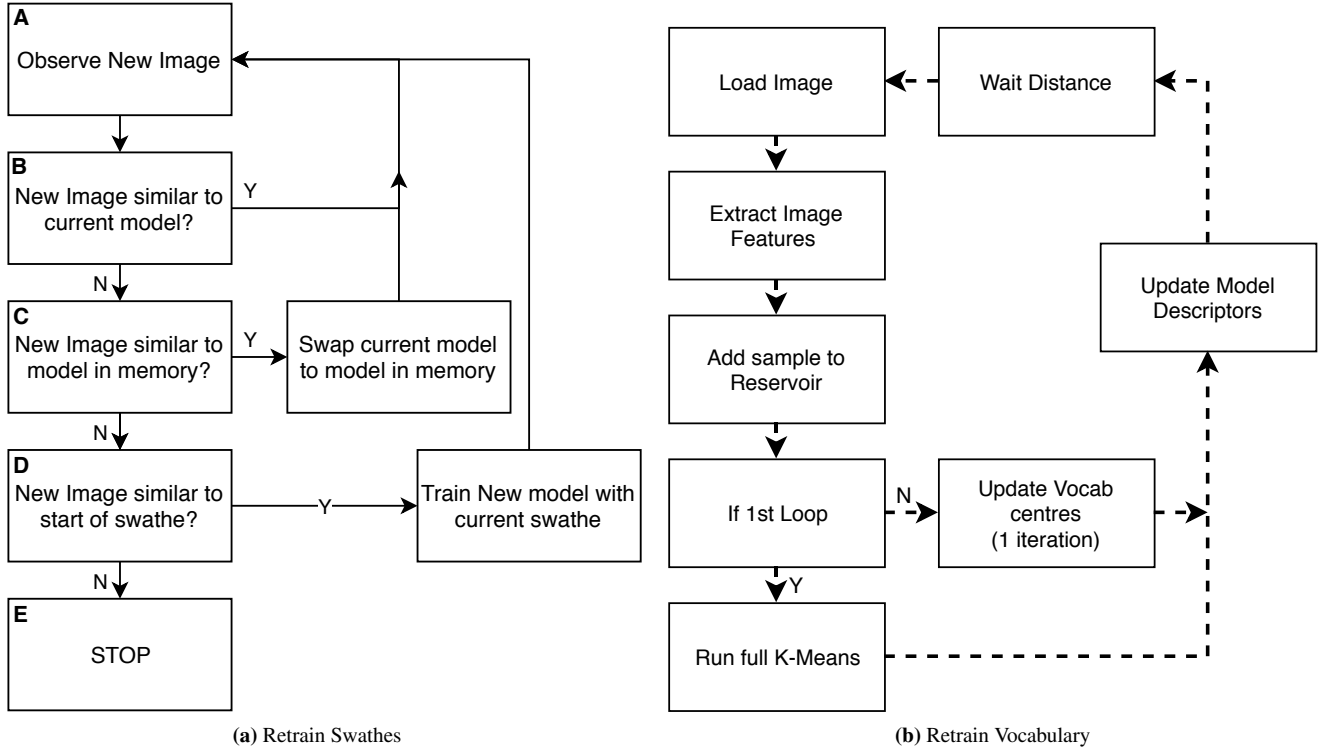


Figure 2.10: Ancillary Tasks used to maintain a representative collection of swathes in memory and to maintain a representative vocabulary.

The Swathe Retraining Loop (Figure 2.10a) is responsible for selecting appropriate models for the current terrain and conditions. The following paragraphs describe each operation labelled in Figure 2.10a.

A A BoW image descriptor is calculated for each new image using the online vocabulary.

B This descriptor $\mathbf{d}_i$ is compared with the descriptor associated with the current model $\mathbf{d}_m$ using the cosine distance for image-to-model distance δ_{im} :

$$\delta_{im} = 1 - \frac{\mathbf{d}_i \cdot \mathbf{d}_m}{\|\mathbf{d}_i\| \|\mathbf{d}_m\|} \quad (2.4)$$

Note the “1−” in the formulation which transforms the standard definition for cosine similarity to cosine distance, this definition decision is arbitrary to the principle and structure of the algorithm. When image descriptors are orthogonal (no overlap between words) $\delta_{im} = 1$, when image descriptors are identical $\delta_{im} = 0$. If the image distance is below a threshold δ_{thresh} , $\delta_{im} \leq \delta_{\text{thresh}}$, the model is deemed suitable and the robot continues without changing models. This threshold determines the algorithms sensitivity to train new models.

C If a new image is dissimilar to the current model, it is compared to all other models in memory. If distance between the current image and a pre-existing model is less than the threshold, this model is selected as the current model without training a new, superfluous model.

D If the current image is dissimilar to the current model and all other models in memory, it is compared with the image from one swathe length behind. If distance between the current image and the start of the swathe, δ_{is} is less than threshold δ_{thresh} , the robot trains a new model using this swathe's data, adds it to memory and continues.

E If $\delta_{is} > \delta_{\text{thresh}}$, the robot stops and the human takes over. Once the human has traversed the area of difficulty, a model is generated from the swathe and is added to memory. This catch block enables the robot to perform primitive introspection.

One important aspect of this framework is that the prior knowledge of the robot, before it begins to operate in an environment is reduced, and where parameters are used they are either non-critical to the algorithm or easily interpretable by a human operator, this has influenced design decisions within the algorithm. For instance, that δ_{thresh} is the same for each image comparison in the Swathe Retraining Loop is a decision made to reduce the overall numbers of parameters. All of the required parameters are listed in Table 2.1. The two prior assumptions made for this algorithm to function are: (1) that areas with lower variance are more drivable (Section 2.3.1) and (2) that areas with similar appearance are similarly drivable (Section 2.3.2). The parameters, these assumptions and the Image Features are the only inputs to the model.

Parameter	Criteria for Choice	Experiment Values
Swathe length	Proportion of Swathe Image covered by training data	20m (Wildcat) 10m (Husky)
Random Forest Trees	Affects accuracy, training times and prediction framerate	10
Image Distance Threshold (δ_{thresh})	Tradeoff between accuracy and number of models trained.	0.6
Vocabulary Length	Affects Vocab Update speed and image similarity effectiveness	100
Reservoir Size	Memory Constraints	100000
Laser Variance Spatial Window ($d_{\text{laser window}}$)	Effects Training Data	0.5m
KNN filter size	Acts as regularizer for Training Data	30
Vertical Variance Threshold ($\theta_{\text{threshold}}$)	Limits the maximum vertical variance in training data.	0.01

Table 2.1: Parameters used in experiments.

2.4 Experiments

2.4.1 Data Collection

This algorithm was tested on three different datasets, collected on two different platforms.

The platforms used were a Bowdler Wildcat 4x4 Vehicle, and a Clearpath Husky All-Terrain Vehicle (ATV). Both were fitted with Point Grey Bumblebee 2 Stereo Cameras, and SICK LMS-151 2D lasers, mounted in a push-broom configuration.

The data was collected in three different places. Example images can be found in Figure 3.1. The ‘Cornbury’ dataset consists of 14 logs (195 km total) collected with the Wildcat platform from woodland in North Oxfordshire. Two different routes were driven, a simpler Cornbury A and a more challenging Cornbury B. This data was collected over a period of two months and contains mostly overcast and sunny conditions.

The ‘Burgess Field’ dataset consists of seven logs (12.5 km total) collected with the Husky in mixed woodland in Oxford. This dataset was collected over three months and has the greatest range of conditions, from overcast, bright sun, and one snow log. Generally this was the hardest dataset, as open terrain and clear skies led to many over-exposed images.

The ‘Uni Parks’ dataset consists of 16 logs (5.2 km total) collected with the Husky in denser woodland in a park in Oxford. As the route is mostly covered by the tree canopy, weather conditions have less impact on the route appearance, but occasionally low sun can cause lens flare in the image. The robot drove primarily over grass and fallen leaves.

There is some disparity in the length of the logs, this is due to platform differences (The Wildcat has a higher top speed than the Husky) and dataset environment differences (The Burgess Field route is simply a longer circuit than University Parks).

Figure 3.1 shows 10 examples from each image, equally spaced through time. Figure 3.2 uses T-SNE clustering to show variation within the data.



Figure 2.11: Different Platforms used for data collection. Left: Bowdler Wildcat. Right: Clearpath Husky.

2.4.2 Benchmarks

In order to test the algorithm, a number of different benchmarks were used. Random Forest Models, with the same parameters as used online were trained on each of the three datasets. These are Burgess Field ‘BF’, University Parks ‘UP’ and Cornbury ‘CB’. The same parameters (primarily the number of trees) were used so that the comparison would be directly between recent data and data from a different environment. It is true that it would have been possible to train more complex models with the extra available data, but harder to attribute any increased performance either to the different environmental conditions or to the increased model complexity. One added benefit of keeping the same Random Forest parameters is that the algorithms have comparable run-time performance. Increasing the model complexity might improve accuracy, but possibly be traded off against runtime speed.

By running every benchmark on every dataset, we would necessarily break the separation between training and test sets, for instance by running the Burgess Field benchmark model on the *2017-11-23-11-07-34-burgess-1* log (which the model was partially trained on). One way to prevent this would have been to train a model on all the Burgess Field data except for this particular log, however it was decided that the data from each log made up a small proportion of the overall training data, and that it was inefficient to train 39 separate ‘leave-one-out’ models on virtually identical datasets, rather than three overall models for each environment.

As well as benchmarking against static random forests, the proposed method is also compared to an algorithm with the online flavour of [Thrun et al., 2006]. In this algorithm, a Random Forest is trained on every swathe, and the most recent model is used for the upcoming path until the next swathe is generated, and newest model trained (labelled ‘Most Recent’ or ‘MR’). This method has the advantage of only using the newest and most relevant data, but is computationally inefficient as the robot expends energy continuously training potentially superfluous models.

Because the algorithm’s data is collected at runtime, it is not straightforward to divide the datasets into training, validation and test sets as is standard practice in traditional datasets. To avoid overfitting, we only validated the proposed algorithm on *2016-04-22-14-18-15-cornbury-off-road-1*, rather than optimising the algorithm for the best possible results across all logs.

2.4.3 Metrics

The prediction error ϵ is defined as

$$\epsilon = \sum_{i=1}^N (y_i - \hat{y}_i)^2, \quad (2.5)$$

the mean squared error between measured and predicted drivability. $y_{i...N}$ is a collection of laser points (usually from a different swathe or from the upcoming image) and prediction $\hat{y}_i = f(x_i, M_b)$, is a function of image features x_i and the chosen model M_b .

To calculate log-wise error, predictions are made on each image according to the algorithm. The drivability of future laser points is calculated with the method in Section 2.3.1. Equation 2.5 is used to calculate the MSE error between the predicted and measured drivability, which is then averaged across each log and presented in Section 2.5.

In order to check that the algorithm is making predictions without reaching the STOP condition, we introduce the metric p_{STOP} is defined as

$$p_{STOP} = \frac{\sum f_{STOP}}{\sum f_{ALL}}, \quad (2.6)$$

the proportion of frames f_{STOP} in which the algorithm reaches the STOP condition (See Figure 2.10) out of all frames f_{ALL} . This metric is used in Figure 2.16.

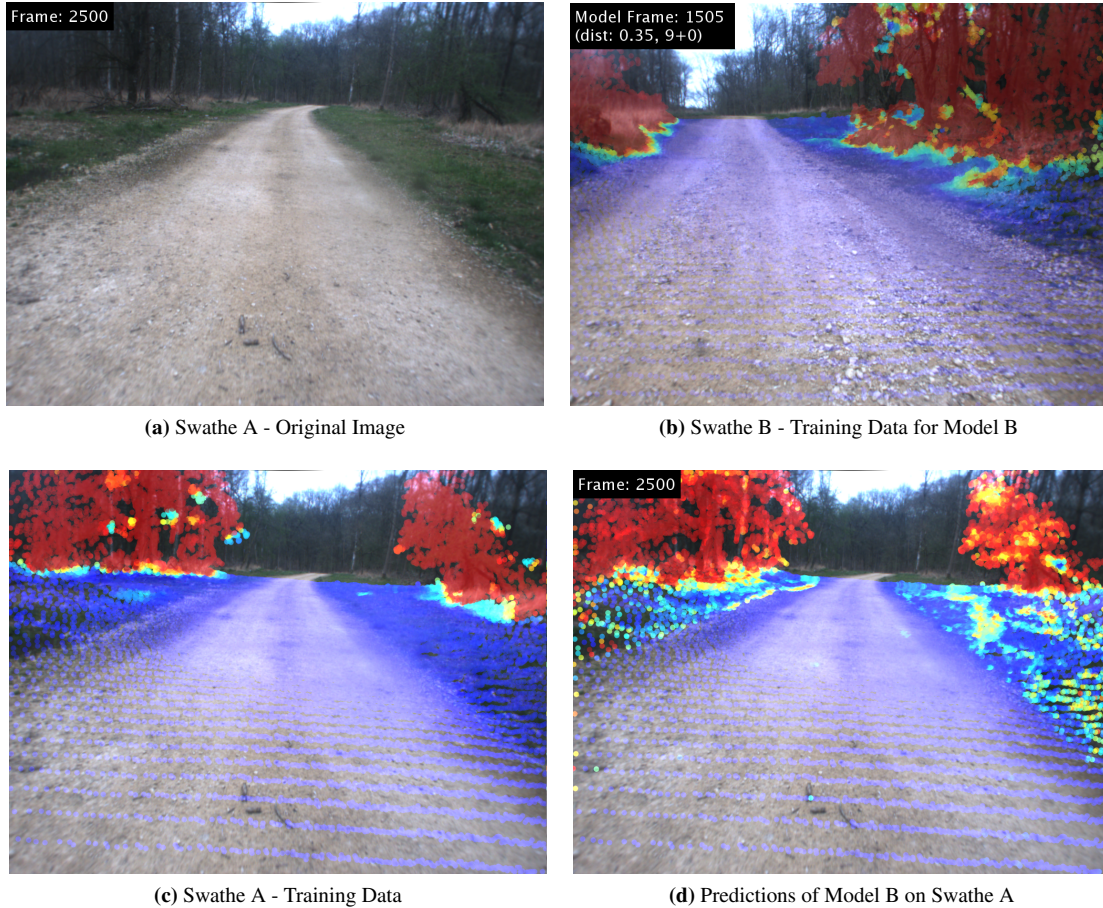


Figure 2.12: Diagram to show how error is calculated. Model B is used to predict the drivability of image points Swathe A. These image points are chosen where the drivability has been measured and is known. The Mean Squared Error is calculated between the Predicted and Measured Values to calculate the Model's performance.

2.5 Results

2.5.1 Preliminary

In order to show that image similarity can be used to select better models, swathes and vocabularies were generated across each log. For each pair of swathes ($\sim 1,000,000$ datapoints), image similarity (Equation 2.3) and predictive error (Equation 2.5) were calculated.

Because each log has its own individual vocabulary, BoW image descriptors cannot be compared across different logs. This meant that all swathe pairs were taken from the same logs.

Figure 2.13 shows samples of error against image similarity, the resulting histogram of all swathe pairs, 50th and 90th percentiles. It shows that the average error increases as images become less similar. However the 90th percentile error, representative of the worst performing models, increases even more. A dense cluster of points is seen in the bottom left corner of the graph, consisting of swathe pairs which are very similar in appearance, with low error.

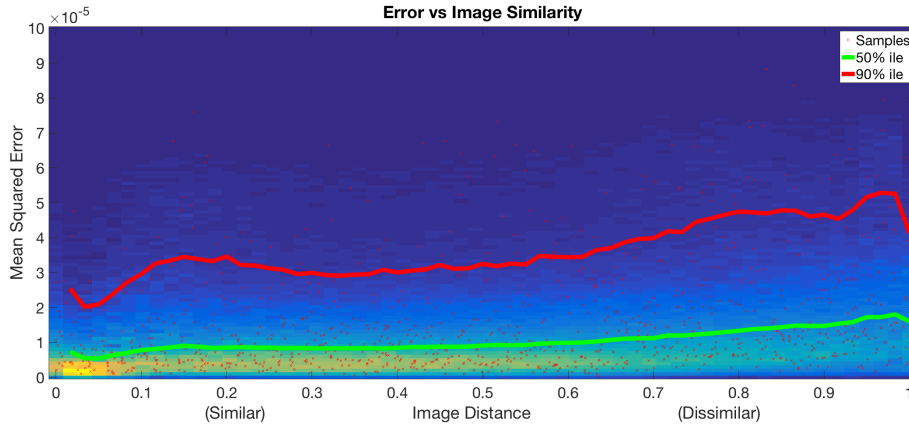


Figure 2.13: Prediction Error (defined in Equation 2.5) of $\sim 1,000,000$ Swathe Pairs across all logs. Green shows the 50th percentile, Red shows 90th percentile error. We see that both the average error and the worst case error increases for swathe pairs with a dissimilar appearance.

2.5.2 Accuracy

The benchmarks and the EBOR algorithm, presented in Section 2.3.4, were run on each log of each dataset and the prediction error calculated for each frame. Figure 2.14 shows the resulting errors for three typical logs.

Table 3.1 shows average error for each log. We see that in some cases the experienced-based approach outperforms the benchmarks, however for the most part the approach doesn't perform as well as the benchmarks.

Figure 2.15 shows the average error by dataset. Unsurprisingly, the fixed Random Forests trained on the particular datasets outperform the fixed Random Forests trained on other datasets. The Cornbury model

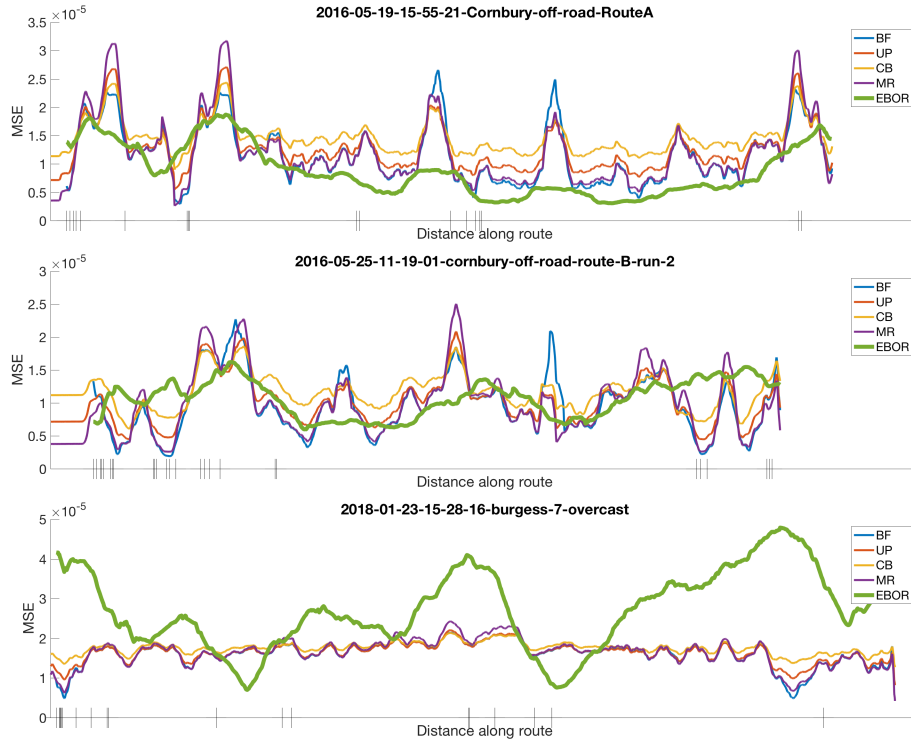


Figure 2.14: Average Error on three different datasets. A 50 frame moving average is applied to all MSE results in order to make Figures clearer. X-axis ticks indicate when EBOR trains a new model, rather than training new models at regular, predefined intervals, the algorithm adapts to when retraining is necessary. Top: EBOR outperforms benchmarks on simple Cornbury environment. Middle: EBOR performance matches the benchmarks. Bottom: EBOR performs significantly worse than the benchmark.

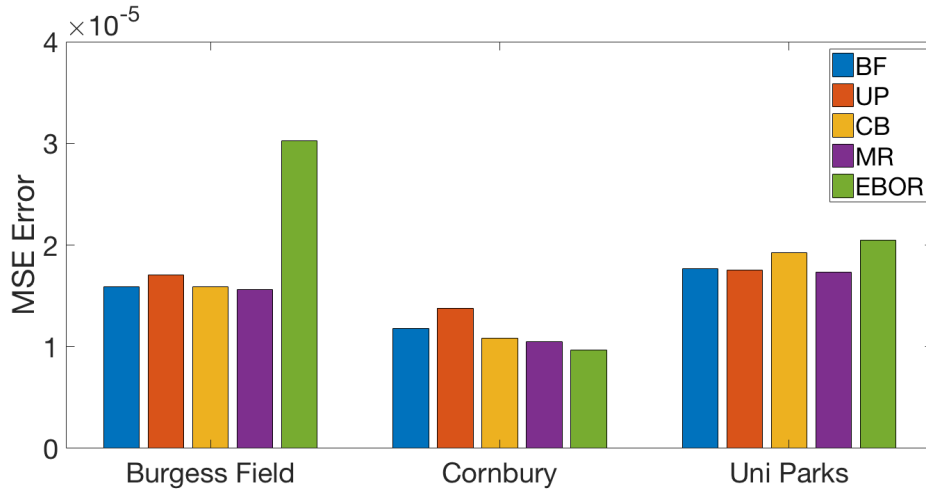


Figure 2.15: Mean Squared Error Averaged by dataset. EBOR is better than the baseline in the Cornbury Environment, commensurate in University Parks but far worse in the Burgess Field environment.

performs well on the Cornbury dataset, the Uni Parks model performs well on the Uni Parks dataset, and so on. However the difference in performance isn't as pronounced as expected. The most-recent benchmark also performs well in all environments. EBOR performs well in the Cornbury dataset, probably given the simplicity of the environment compared with the other datasets. Figure 2.19 shows examples of the EBOR algorithm failing.

2.5.3 Algorithmic Efficiency

Table 3.2 shows the number of models trained by the EBOR and Most Recent Algorithms respectively. Clearly EBOR is far more efficient in all cases, although no effort was made to streamline the Most Recent Benchmark. The thousands of models trained by using the Most Recent benchmark are almost certainly superfluous but give an indication of the performance that might be possible if computational efficiency of continuously training models is not a factor in designing an algorithm.

One other indication of the algorithms performance is the p_{STOP} percentage. This indicates the percentage of the logs for which EBOR couldn't find an appropriate model, either within memory or by inspecting the most recent swathe. The percentage is low for most models, with a few exceptions. 'burgess-2' (26%), 'uni-parks' (13.9%), 'uni-parks-offroad-6' (13.9%) and 'uni-parks-offroad-12' (14.3%) were the highest fractions. Generally speaking these logs were shorter than their counterparts, suggesting that perhaps the algorithms were still burning in. p_{STOP} is equivalent to primitive introspection - rather than continue with a model that the system cannot guarantee performance, EBOR decides not to make a prediction. In these situations, the robot would hand back control to a human operator to navigate the area of uncertainty. Figure ?? shows p_{STOP} as each log progresses. It demonstrates that over time the algorithm makes more predictions, supporting the hypothesis that shorter logs have higher p_{STOP} values due to early uncertainty.

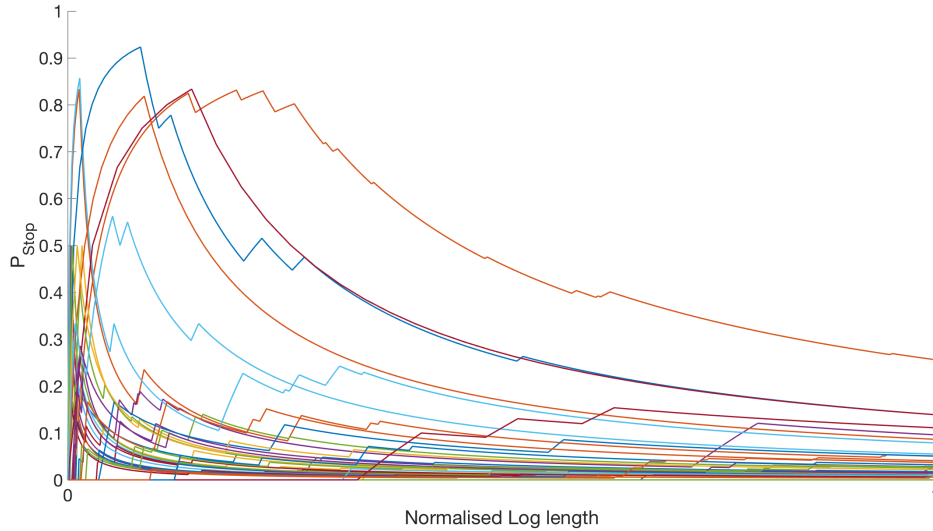


Figure 2.16: The proportion of frames, p_{STOP} (Equation 2.6), for which the algorithm is unable to find an appropriate model, plotted cumulatively as each log progresses. The x-axis is normalised by the length of the log for clearer plotting. Each uptick indicates a period when EBOR refrains from making predictions. Initially p_{STOP} is high for all logs. However, over time the model becomes more confident as it curates a collection of swathes and is more likely to find a suitable model to make a prediction.

2.5.4 Timing

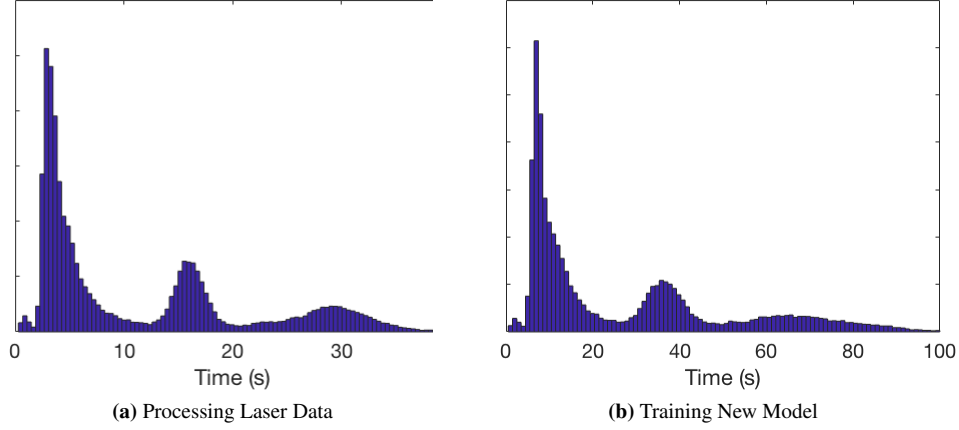


Figure 2.17: Histograms for Time Taken to Process laser data and train models whilst generating results in Table 3.1. Left: Processing Laser Data includes combining successive pointclouds and calculating Vertical Variance with Equation 2.1. Right: Time Taken to train new Random Forest Model in Equation 2.2.

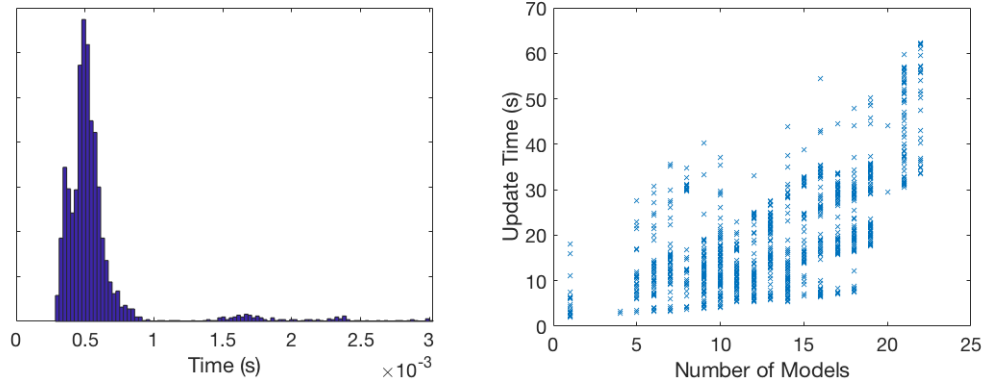


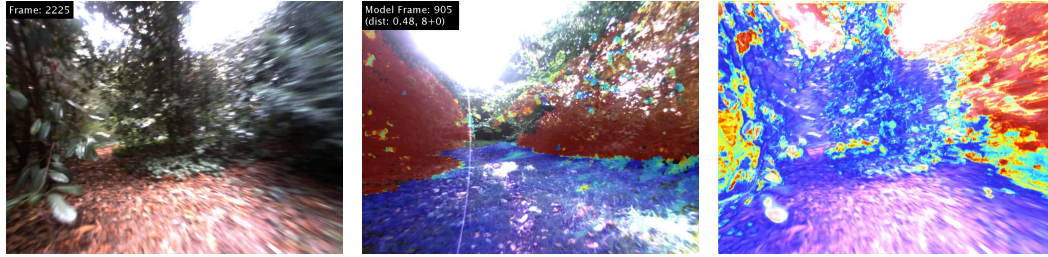
Figure 2.18: Left: Time taken to find and load a suitable model from memory. (Note the order of magnitude). Right: Time taken to update descriptors of each swathe 2.10b. Time taken to update descriptors is linear with the number of model images that need updating.

The first thing to mention when discussing the timing of the algorithm is that code was not optimised to run as fast as possible. Firstly it was implemented in MATLAB, rather than a compiled language such as C++. Secondly, much of the algorithm is able to run in parallel, such as training Random Forest trees in parallel or updating descriptors for multiple models. However, due to the way in which the results were generated, the timing results are presented for single-threaded ‘serial’ processes.

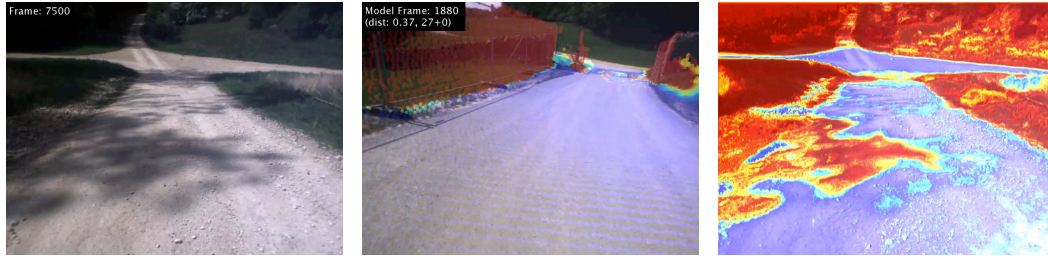
Figure 2.17a shows the time taken to process laser data to generate swathes, according to the method presented in Section 2.3.1. Figure 2.17b shows the time taken to train a Random Forest model using the processed laser data. There is a strong correlation between these two graphs, because the greatest factor in processing and computation time is the number of datapoints that have to be processed for each swathe, or trained on in each model. This also explains the multimodal distribution of the graph. Each

mode corresponds to a different dataset, a different platform speed and a different average number of laser points in each swathe. The Wildcat traverses each swathe faster, and as the push-broom laser runs at a fixed frequency, each swathe contains fewer training points. In the University Parks dataset, the Husky travels most slowly, resulting in a longer time to traverse each swathe, and a greater number of datapoints in each swathe. Table 3.2 shows the average time taken to train each random forest, t_{RF} , and the average time to process each swathe t_{laser} for each log, as well as the average speed of the robot, v_{mean} . There is a clear correlation between high platform speeds and low training times, even despite the different swathe lengths in Table 2.1. As discussed above, both training random forests and processing swathes are naturally parallelisable.

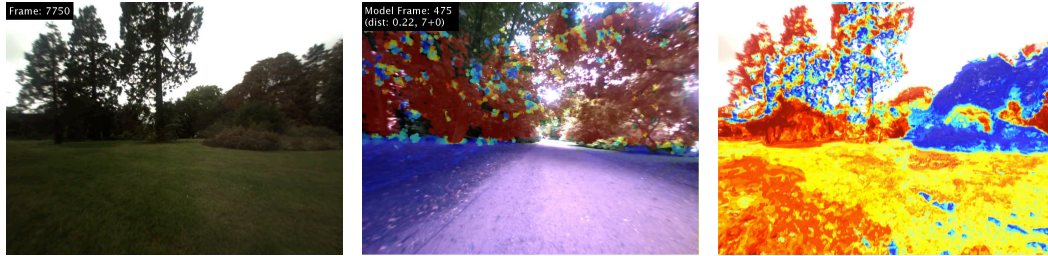
Figure 2.18a shows the time taken to retrieve a suitable model from memory. In practice this involves calculating the distance between the current image and all in memory using Equation 2.4. Because all swathe image descriptors are updated ahead of time, this is a very fast operation. Figure 2.18b shows this update process. This is a computationally intensive process, however it runs in the background to keep the vocabulary up to date and is not essential to the core loop. In these results the model descriptors were updated serially, leading to the $\mathcal{O}(n_{\text{models}})$ computation time. This operation would be straightforward to implement in parallel, alternatively the vocabulary could be updated less often, reducing this computational load. However, this was not explored in this project.



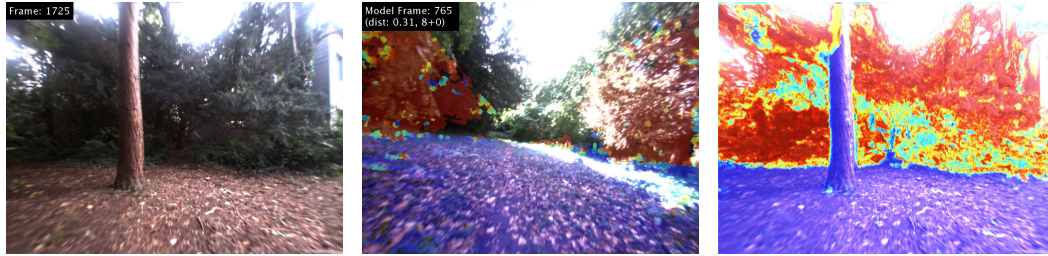
(d) Model trained on flat green grass identifies green vegetation as drivable.



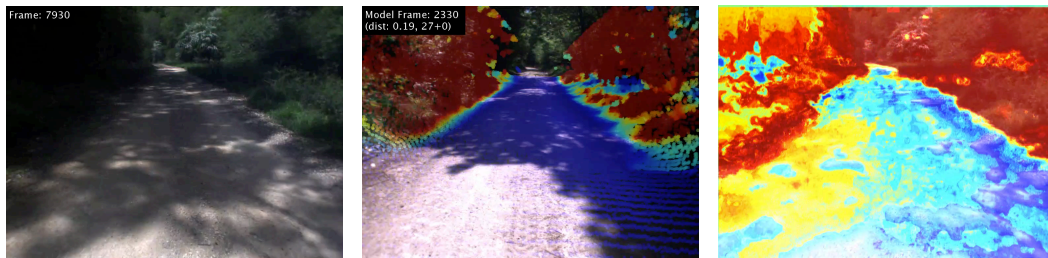
(e) Model trained in absence of shadows identifies shadows across path as undrivable.



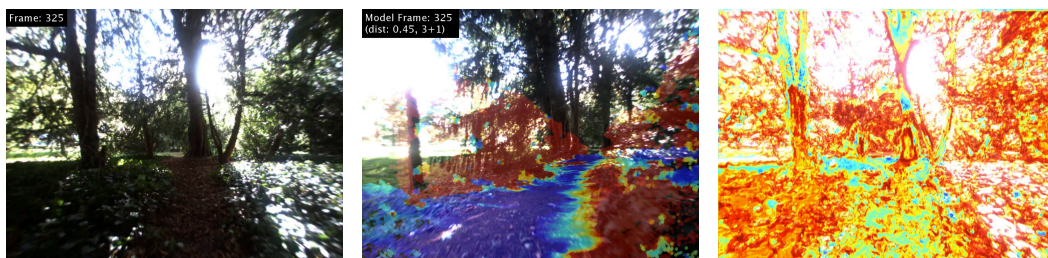
(f) Algorithm loads inappropriate road swathe in grassy environment.



(g) Tree predicted as drivable as similar colour to ground in training swathe.



(h) Despite training swathe containing shadows, model is still unable to identify it as drivable path, although it correctly predicts it as more drivable than surrounding vegetation.



(i) Model completely unable to detect drivable path, possibly due to poor lighting.

Figure 2.19: Examples of failure cases with EBOR algorithm. Left: Current Image. Centre: Current Model in memory. Right: Resulting Drivability Predictions.

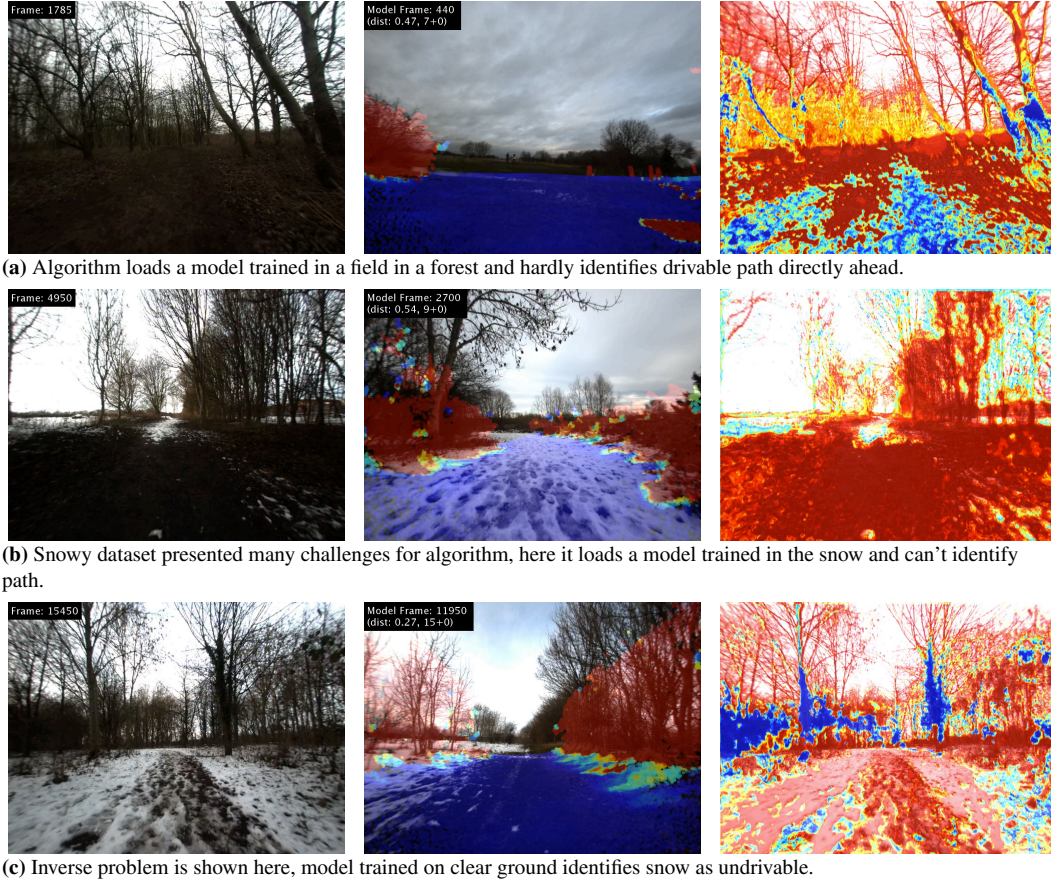


Figure 2.19: Examples of failure cases with EBOR algorithm. Left: Current Image. Centre: Current Model in memory. Right: Resulting Drivability Predictions.

2.5.5 Discussion

Clearly the performance of the EBOR algorithm is below the standard set by the benchmark. Some potential reasons can be ruled out quickly. For instance, the discrepancy in performance could be due to discrepancies in the data collection. The Wildcat (Cornbury) runs (which the algorithm performs well on) are much longer than the Husky ones. Is it possible that the EBOR algorithm doesn't have long enough to be established on the shorter runs, akin to the 'burn in' phase of Monte Carlo sampling? This can be discounted for two primary reasons. Firstly because EBOR performance decreases on the Cornbury B logs (in the bottom half of Table 3.1), which are as long as Cornbury A. The only difference between them is scene complexity. Secondly, because performance isn't seen to increase towards the end of logs. For instance, the 'burgess-7-overcast' log (Bottom of Figure 2.14) shows that performance is worse at the end of the log than at the start.

The reasons for EBOR under-performing on logs of greater scene complexity are more nuanced and harder to rectify. One of the core assumptions of the algorithm is that selecting models using image similarity leads to lower errors. While Figure 2.13 shows this to be the case theoretically, the correlation isn't strong enough in practice, there are still plenty of outliers.

Perhaps the approach of using vertical variance isn't good enough. Consider the problem of misla-

belled data. For instance in the event that Visual Odometry failed, or a dynamic obstacle is captured by the laser but not by the camera. For the static Random Forest benchmarks, mislabelled data is negated by greater amounts of correctly labelled data. For the ‘Most Recent’ benchmark, an incorrectly labelled swathe is quickly replaced by retrained models. However, if EBOR trains a model with a mislabelled swathe, that bad model may be used repeatedly and accrue large errors.

The algorithm incorporates no element of introspection which might be able to identify poor swathes and models. Image similarity is the only factor used to decide the suitability of different swathes.

It would be possible to add extra introspection modules or logic to the algorithm that could increase performance. Perhaps multiple models should be used for prediction and their n predictions averaged. Perhaps models should be evaluated on swathes generated every x metres, or with probability p and their accuracy accounted for when choosing swathes. Not only would these additions increase the computational cost of the algorithm and undermine potential computational savings, but to do so would nullify the main motivations and attractions of the algorithm, namely simplicity and interpretability.

The number of different tweaks, additions and modifications to alter the algorithm are great. However, adding layers of algorithmic complexity and hand picking the associated model parameters makes the algorithm less clear and potentially less applicable to other environments. The more parameters contained within the algorithm, the more likely it is that we are overfitting to these particular datasets.

2.6 Conclusions

The algorithm assumes that Image Similarity is the best way to identify models that are likely to succeed. However in practice this is not the case. Metric learning [Yang and Jin, 2006, Bellet et al., 2013] is an area of Machine Learning which learns a distance function. In this case it could be used to learn a mapping between source and target swathe images that would learn to predict one model’s performance on another. This would be an alternative to using cosine distance and BoW descriptors. Given enough training data it would be able to select better, more suitable models at runtime. However, significant difficulties would have to be overcome. One nice feature of using cosine distance between BoW descriptors is that laser data only needs to be processed when training new models (which is a significant computational cost - see Section 2.5.4). If metric learning were employed, computational cost could increase severely, in processing laser data to train the distance metric as well as training the distance metric itself. This process would add extra arbitrary parameters which might be difficult to tune or understand for an operator.

If more laser data was available to the algorithm by processing more swathes, it would be possible to select models based on their accuracy predicting this data. Rather than selecting models with the most similar images, models with a proven track record would be selected more often than models which were

prone to higher errors. However, not only might this take longer to burn in and acquire the necessary samples, but it would undermine the incentive to train models in specific areas. If all models are evaluated on all swathes, the one with lowest overall error will be the model with the lowest overall error, not necessarily the specialist for the current environment.

If a generic overall model was deemed preferable to a set of models trained in different environments, an Online Random Forest [Lakshminarayanan et al., 2014, Leistner and Bischof, 2009] could be trained on a continual stream of laser data. However, it's not clear whether this would be more any more computationally efficient than the continuous training of the 'Most Recent' benchmark. It would still need to have some idea of when to retrain efficiently, and would still need to process laser data.

The other alternative would be to use Deep Learning in some form or other. The fact that the Static Random Forest benchmarks (CB, UP, BF) had comparable results on the other datasets (Figure 2.15) suggests that these datasets were similar enough that information could be shared between them. Initially Deep Neural Networks (DNNs) were overlooked as there were no guarantees that they could be used in different environments, similar to Random Forests. However, if the Random Forest models can perform in other datasets then potentially the Deep Learning Models could as well. Training on one dataset and testing on another may well be 'good enough' for purpose. One advantage of using DNNs would be their ability to use textural information and contextual cues without having to hand craft particular image features as in Section 2.3.2.

One disadvantage of using traditional DNNs is that they are poor at representing uncertainty [Kendall and Gal, 2017], and due to the large number of parameters they also require very large training datasets or they can be susceptible to over-fit. One antidote might be to use Bayesian Neural Networks [Kendall et al., 2015] which provide a framework for capturing uncertainty within the data. By learning the Epistemic uncertainty [Kendall and Gal, 2017] the uncertainty due to a lack of data, the algorithm could learn to only operate in environments close to the training datasets. An algorithm could be developed in which a Bayesian DNN was used until uncertainty passed a threshold. Once this happened, it could be fine-tuned using online swathe data, naively or with a more complicated Active Learning method, for instance [Gal et al., 2017], or the 'Most Recent' Random Forest model could be used as an interim model until the DNN architecture was certain again. This is a similar approach to [Richter and Roy, 2017], which switches from a more advanced and complicated Neural Planner to a simpler, more reliable controller when the Neural Planner is uncertain.

In this work we show that using real-world measurements is a convenient method for generating large quantities of labelled training data. We use measured energy consumption and publically available maps to extrapolate consumption to unseen, previously unexplored map routes. By using a suitable probabilistic

model we can account for inherent variance and noise in the energy readings. These predictions are useful for high-level route-planning decisions but are reliant on the quality of map annotations. When map annotations and prior information about an environment are poor we explore different methods to assess drivability. We use a laser to label the drivability of terrain in a swathe and propose a new method to use this data in an online setting, this method performs better than the benchmarks in simpler environments such as Cornbury, but shows flaws in more complicated environments.

Chapter 3

Appendices

This section contains Figures and graphs too large for earlier Sections. Figure 3.1 shows example images from each of the datasets. Figure 3.2 clusters each of these by image similarity to show that the recorded data is not homogenous and that appearance differences are clear between them. Table 3.1 contains the full MSE results for the EBOR algorithm and it's benchmarks in each of the datasets. Table 3.2 contains miscellaneous information about each log.



Figure 3.1: Images equally spread through each dataset.



Figure 3.1: Images equally spread through each dataset.



Figure 3.1: Images equally spread through each dataset.

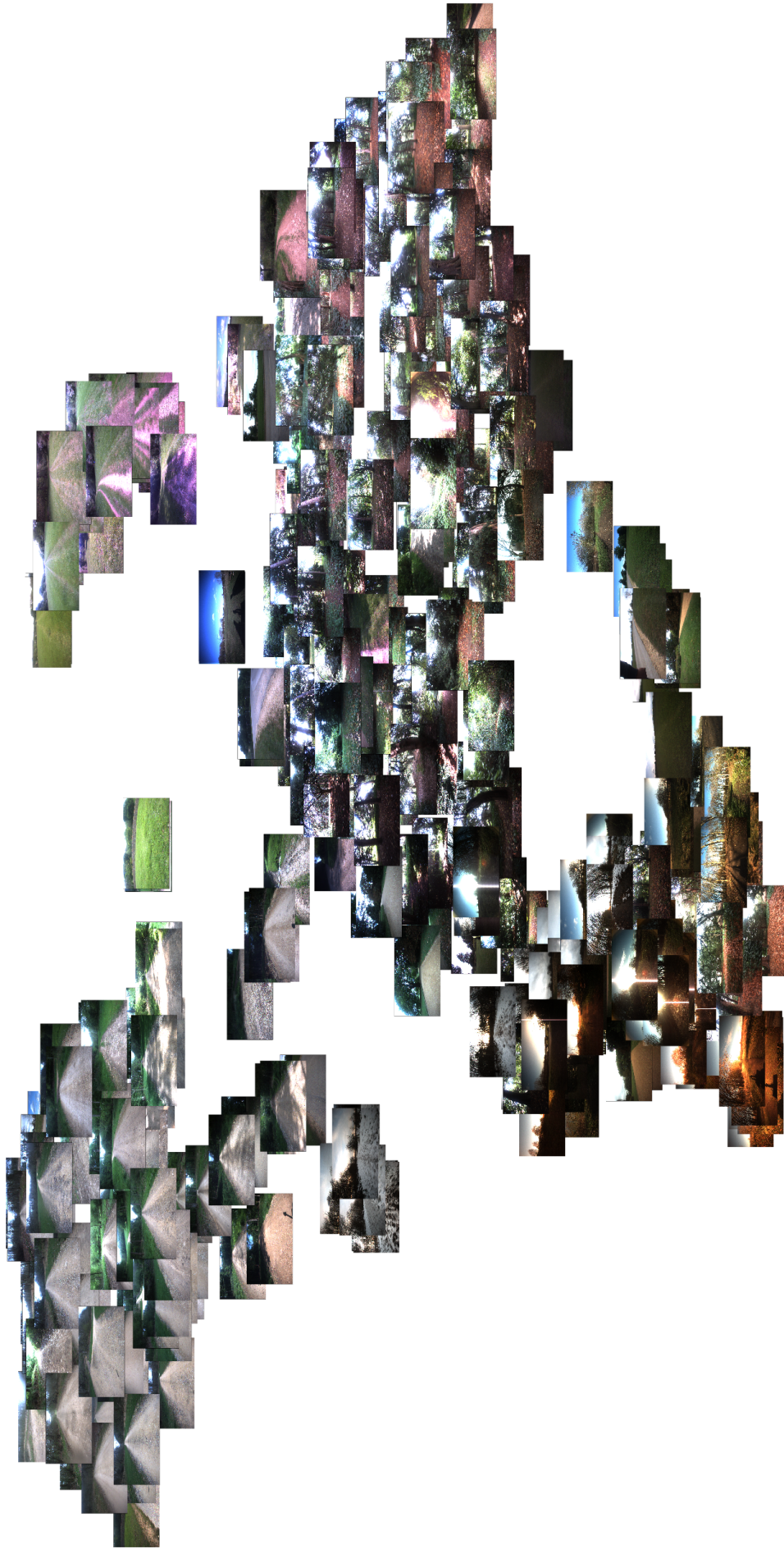


Figure 3.2: T-SNE Clustering of representative images from all datasets. Using the ‘global’ vocabulary from Section 2.3.3, BoW descriptors were calculated for each image in Figure 3.1. Clustering was performed with T-SNE [van der Maaten and Hinton, 2008] to show the heterogeneous nature of the different datasets. Different clusters include Cornbury A (Top Left), Burgess Field Snow (Middle Left), Burgess Field (Bottom), Cornbury B (Top Right), University Parks (Middle Right).

Log Name	Log Type	BF	UP	CB	MR	EBOR
2017-11-23-11-07-34-burgess-1	BF	0.146	0.159	0.145	0.142	0.332
2017-11-23-11-52-30-burgess-2	BF	0.131	0.153	0.119	0.113	0.189
2017-12-01-10-26-31-burgess-3	BF	0.177	0.182	0.184	0.171	0.296
2017-12-01-10-56-34-burgess-4	BF	0.173	0.180	0.178	0.170	0.285
2017-12-12-14-42-12-burgess-5-snow	BF	0.157	0.169	0.157	0.167	0.510
2018-01-18-10-07-49-burgess-6-bright	BF	0.164	0.175	0.165	0.169	0.234
2018-01-23-15-28-16-burgess-7-overcast	BF	0.164	0.175	0.165	0.161	0.273
2016-04-22-14-18-15-combury-off-road-1	CB	0.116	0.143	0.098	0.091	0.065
2016-04-22-14-29-14-combury-off-road-2	CB	0.114	0.142	0.095	0.085	0.048
2016-04-28-12-09-56-combury-off-road-RouteA	CB	0.115	0.142	0.098	0.093	0.086
2016-04-28-13-28-13-combury-off-road-RouteA	CB	0.115	0.142	0.096	0.093	0.070
2016-05-19-14-12-00-combury-off-road-routeA	CB	0.119	0.140	0.109	0.107	0.086
2016-05-19-14-49-52-combury-off-road-routeA	CB	0.121	0.141	0.112	0.108	0.090
2016-05-19-15-55-21-combury-off-road-RouteA	CB	0.129	0.146	0.121	0.117	0.088
2016-05-23-15-23-10-combury-off-road-route-A	CB	0.136	0.151	0.132	0.129	0.116
2016-05-23-15-58-17-combury-off-road-route-b	CB	0.108	0.122	0.104	0.103	0.123
2016-05-23-16-35-15-combury-off-road-route-a	CB	0.128	0.146	0.122	0.123	0.107
2016-05-25-10-42-04-combury-off-road-route-B-run-1	CB	0.108	0.121	0.107	0.103	0.115
2016-05-25-11-19-01-combury-off-road-route-B-run-2	CB	0.101	0.116	0.097	0.097	0.106
2016-05-26-11-00-31-combury-route-b-run-1	CB	0.110	0.125	0.105	0.102	0.115
2016-05-26-12-44-34-combury-route-a-run-2	CB	0.130	0.147	0.124	0.120	0.135
2017-09-05-11-23-48-uni-parks	UP	0.133	0.156	0.118	0.139	0.229
2017-09-05-12-07-01-uni-parks-offroad	UP	0.183	0.185	0.195	0.188	0.185
2017-09-05-12-14-04-uni-parks-offroad-2	UP	0.147	0.164	0.141	0.129	0.285
2017-09-12-12-29-11-uni-parks-offroad-3	UP	0.184	0.178	0.205	0.175	0.214
2017-09-12-12-47-04-uni-parks-offroad-4	UP	0.183	0.180	0.201	0.178	0.219
2017-09-13-14-58-51-uni-parks-offroad-5	UP	0.161	0.160	0.176	0.178	0.173
2017-09-13-15-54-51-uni-parks-offroad-6	UP	0.199	0.191	0.221	0.185	0.097
2017-09-14-14-17-47-uni-parks-offroad-7	UP	0.177	0.165	0.203	0.162	0.171
2017-09-14-14-35-05-uni-parks-offroad-8	UP	0.188	0.181	0.209	0.180	0.236
2017-09-18-11-09-46-uni-parks-offroad-9	UP	0.196	0.183	0.224	0.184	0.179
2017-09-18-11-27-41-uni-parks-offroad-10	UP	0.178	0.176	0.194	0.178	0.172
2017-09-22-15-09-08-uni-parks-offroad-11	UP	0.180	0.176	0.198	0.185	0.203
2017-09-25-13-37-57-uni-parks-offroad-11b	UP	0.189	0.180	0.213	0.182	0.190
2017-09-25-14-56-01-uni-parks-offroad-12	UP	0.182	0.182	0.197	0.180	0.296
2017-09-28-15-49-49-uni-parks-offroad-13-sun	UP	0.181	0.175	0.203	0.180	0.235
2018-01-10-09-44-55-uni-parks-offroad-17	UP	0.172	0.172	0.185	0.172	0.193
Average	-	0.151	0.172	0.185	0.172	0.193

Table 3.1: Mean Squared Error ($\times 10^{-5}$) for Static Models trained on Burgess Field (BF), University Parks (UP) and Cornbury (CB), Most Recent Swathe Algorithm (MR) and Experience Based Off-Road (EBOR)

Log Name	Length (m)	v_{mean} (m/s)	n_{models} (EBOR)	n_{models} (MR)	p_{stop} %	t_{RF} (s)	t_{laser} (s)
2017-11-23-11-07-34-burgess-1	1030	0.47	4360	19	1.81	64.77	28.34
2017-11-23-11-52-30-burgess-2	446	0.46	1956	22	26.04	65.62	27.58
2017-12-01-10-26-31-burgess-3	1440	0.83	3382	10	0.62	42.56	19.24
2017-12-01-10-56-34-burgess-4	1301	0.82	2971	10	1.54	35.86	16.69
2017-12-12-14-42-12-burgess-5-snow	2717	0.81	4198	3	0.00	45.09	18.62
2018-01-18-10-07-49-burgess-6-bright	2705	0.85	4510	19	0.64	38.92	16.85
2018-01-23-15-28-16-burgess-7-overcast	2911	0.86	6748	19	0.82	36.03	15.68
2016-04-22-14-18-15-combury-off-road-1	3753	6.48	2297	9	1.61	9.41	4.27
2016-04-22-14-29-14-combury-off-road-2	10004	5.84	2938	10	0.58	10.24	4.65
2016-04-28-12-09-56-combury-off-road-RouteA	10064	6.80	5603	8	1.10	9.97	4.32
2016-04-28-13-28-13-combury-off-road-RouteA	10048	5.98	6043	11	2.83	10.93	4.93
2016-05-19-14-12-00-combury-off-road-routeA	20138	6.57	5672	14	1.60	11.73	5.02
2016-05-19-14-49-52-combury-off-road-routeA	20096	6.60	5624	16	1.17	11.05	4.83
2016-05-19-15-55-21-combury-off-road-RouteA	22425	6.02	6171	18	0.90	10.26	4.60
2016-05-23-15-23-10-combury-off-road-route-A	19897	6.97	5538	12	1.49	12.24	5.54
2016-05-23-15-58-17-combury-off-road-route-b	11683	3.67	6098	14	5.05	21.85	8.91
2016-05-23-16-35-15-combury-off-road-route-a	20171	6.46	5734	16	0.89	9.43	4.13
2016-05-25-10-42-04-combury-off-road-route-B-run-1	8879	3.74	5852	21	9.60	34.52	15.54
2016-05-25-11-19-01-combury-off-road-route-B-run-2	10754	4.51	4744	26	2.90	13.50	5.26
2016-05-26-11-00-31-combury-route-b-run-1	7261	3.45	3250	18	7.87	21.11	8.61
2016-05-26-12-44-34-combury-route-a-run-2	20264	7.76	4177	15	1.06	7.50	3.29
2017-09-05-11-23-48-uni-parks	190	0.38	781	10	13.89	60.14	27.91
2017-09-05-12-07-01-uni-parks-offroad	140	0.42	562	6	8.74	71.93	31.58
2017-09-05-12-14-04-uni-parks-offroad-2	420	0.49	1610	7	1.29	58.81	25.88
2017-09-12-12-29-11-uni-parks-offroad-3	422	0.47	1683	10	1.44	73.98	29.78
2017-09-12-12-47-04-uni-parks-offroad-4	257	0.45	1034	10	2.51	75.86	31.19
2017-09-13-14-58-51-uni-parks-offroad-5	125	0.34	582	4	1.01	118.30	51.25
2017-09-13-15-54-51-uni-parks-offroad-6	78	0.40	207	4	13.89	70.26	30.87
2017-09-14-14-17-47-uni-parks-offroad-7	114	0.43	327	8	5.26	77.26	30.56
2017-09-14-14-35-05-uni-parks-offroad-8	231	0.46	704	7	3.79	77.82	32.56
2017-09-18-11-09-46-uni-parks-offroad-9	307	0.44	950	8	2.19	79.44	33.21
2017-09-18-11-27-41-uni-parks-offroad-10	428	0.48	1290	9	2.81	71.32	30.82
2017-09-22-15-09-08-uni-parks-offroad-11	728	0.47	1863	16	2.76	75.18	30.96
2017-09-25-13-37-57-uni-parks-offroad-11b	389	0.46	1221	8	5.58	111.82	45.01
2017-09-25-14-56-01-uni-parks-offroad-12	62	0.27	344	7	14.29	237.48	96.13
2017-09-28-15-49-49-uni-parks-offroad-13-sun	440	0.44	1435	8	3.24	113.60	48.10
2018-01-10-09-44-55-uni-parks-offroad-17	872	0.79	2148	19	4.11	42.60	18.18

Table 3.2: Table of Timing and Miscellaneous Results. Columns are: Log Length, Average Speed of Robot during traversal v_{mean} , number of models n_{models} generated by algorithm for Most Recent (MR) and EBOR algorithms, proportion of time spent by EBOR algorithm in stop condition p_{stop} Equivalent to E in Figure 2.10a, average time taken to train new Random Forests t_{RF} , average time taken to process new laser data t_{laser} .

Bibliography

- [eur, 2013] (2013). The European Pedestrian Assistant 2.0. [\url{http://europa2.informatik.uni-freiburg.de/}](http://europa2.informatik.uni-freiburg.de/).
- [Adelson et al., 1984] Adelson, E. H., Anderson, C. H., Bergen, J. R., Burt, P. J., and Ogden, J. M. (1984). Pyramid methods in image processing. *Engineer*, 29(October):33–41.
- [Altmann et al., 2010] Altmann, A., Tolo\csi, L., Sander, O., and Lengauer, T. (2010). Permutation importance: a corrected feature importance measure. *Bioinformatics*, 26(10):1340–1347.
- [Arandjelovic et al., 2017] Arandjelovic, R., Gronat, P., Torii, A., Pajdla, T., and Sivic, J. (2017). NetVLAD: CNN architecture for weakly supervised place recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 5297–5307.
- [Bajracharya et al., 2008] Bajracharya, M., Tang, B., Howard, A., Turmon, M., and Matthies, L. (2008). Learning long-range terrain classification for autonomous navigation. *Proceedings - IEEE International Conference on Robotics and Automation*, (July):4018–4024.
- [Barnes et al., 2017] Barnes, D., Maddern, W., and Posner, I. (2017). Find Your Own Way : Weakly-Supervised Segmentation of Path Proposals for Urban Autonomy. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 203–210.
- [Basu and Meckesheimer, 2007] Basu, S. and Meckesheimer, M. (2007). Automatic outlier detection for time series: an application to sensor data. *Knowledge and Information Systems*, 11(2):137–154.
- [Bellet et al., 2013] Bellet, A., Habrard, A., and Sebban, M. (2013). A survey on metric learning for feature vectors and structured data. *arXiv preprint arXiv:1306.6709*.
- [Berczi and Barfoot, 2016] Berczi, L. P. and Barfoot, T. D. (2016). It’s like déjà vu all over again: Learning place-dependent terrain assessment for visual teach and repeat. *IEEE International Conference on Intelligent Robots and Systems*, 2016-Novem:3973–3980.
- [Berczi et al., 2015] Berczi, L. P., Posner, I., and Barfoot, T. D. (2015). Learning to assess terrain from human demonstration using an introspective Gaussian-process classifier. *Proceedings - IEEE International Conference on Robotics and Automation*, 2015-June(June):3178–3185.
- [Bibuli et al., 2007] Bibuli, M., Caccia, M., and Lapierre, L. (2007). Path-following algorithms and experiments for an autonomous surface vehicle. *IFAC Proceedings Volumes (IFAC-PapersOnline)*, 7(PART 1):81–86.
- [Binois et al., 2018] Binois, M., Gramacy, R. B., and Ludkovski, M. (2018). Practical Heteroscedastic Gaussian Process Modeling for Large Simulation Experiments Practical Heteroscedastic Gaussian Process Modeling for Large ABSTRACT. 8600.
- [Brateman et al., 2006] Brateman, J., Xian, C., and Lu, Y.-H. (2006). Energy-Efficient Scheduling for Autonomous Mobile Robots. In *Very Large Scale Integration, 2006 IFIP International Conference on*, pages 361–366. IEEE.
- [Broderick et al., 2014] Broderick, J. A., Tilbury, D. M., and Atkins, E. M. (2014). Characterizing Energy Usage of a Commercially Available Ground Robot: Method and Results. *Journal of Field Robotics*, 31(3):441–454.
- [Brooks, 2009] Brooks, C. (2009). Visual detection of novel terrain via two-class classification. *of the 2009 ACM symposium on Applied*, 3262:1145–1150.

- [Brooks and Iagnemma, 2012] Brooks, C. A. and Iagnemma, K. (2012). Self-supervised terrain classification for planetary surface exploration rovers. *Journal of Field Robotics*, 29(3):445–468.
- [Carroll, 2012] Carroll, C. B. C. W. S. (2012). Impact of driving characteristics on electric vehicle energy consumption and range. (January 2011):29–35.
- [Churchill and Newman, 2013] Churchill, W. and Newman, P. (2013). Experience-based navigation for long-term localisation. *International Journal of Robotics Research*, 32(14):1645–1661.
- [Cummins and Newman, 2008] Cummins, M. and Newman, P. (2008). FAB-MAP: Probabilistic localization and mapping in the space of appearance. *The International Journal of Robotics Research*, 27(6):647–665.
- [Cummins and Newman, 2011] Cummins, M. and Newman, P. (2011). Appearance-only SLAM at large scale with FAB-MAP 2.0. *The International Journal of Robotics Research*, 30(9):1100–1123.
- [Dahlkamp et al., 2006] Dahlkamp, H., Kaehler, A., Stavens, D., Thrun, S., and Bradski, G. (2006). Self-supervised Monocular Road Detection in Desert Terrain. *Proc of Robotics Science and Systems RSS*.
- [De Brabandere et al., 2016] De Brabandere, B., Jia, X., Tuytelaars, T., and Van Gool, L. (2016). Dynamic Filter Networks. pages 1–14.
- [Elnahrawy and Nath, 2003] Elnahrawy, E. and Nath, B. (2003). Cleaning and querying noisy sensors. In *Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*, pages 78–87. ACM.
- [Fallah et al., 2013] Fallah, S., Yue, B., Vahid-Araghi, O., and Khajepour, A. (2013). Energy Management of Planetary Rovers Using a Fast Feature-Based Path Planning and Hardware-in-the-Loop Experiments. *Vehicular Technology, IEEE Transactions on*, 62(6):2389–2401.
- [Gal et al., 2017] Gal, Y., Islam, R., and Ghahramani, Z. (2017). Deep Bayesian Active Learning with Image Data. *Nips*.
- [Garcia-garcia et al., 2018] Garcia-garcia, A., Orts-escolano, S., Oprea, S., Villena-martinez, V., Martinez-gonzalez, P., and Garcia-rodriguez, J. (2018). A survey on deep learning techniques for image and video semantic segmentation. *Applied Soft Computing Journal*, 70:41–65.
- [Goldberg et al., 1997] Goldberg, P. W., Williams, C. K. I., and Bishop, C. M. (1997). Regression with input-dependent noise: A Gaussian process treatment. *Advances in neural information processing systems*, 10:493–499.
- [Guisti et al., 2016] Guisti, A., Guzzi, J., Circesan, D. C., He, F.-L., Rodriguez, J. P., Fontana, F., Faessler, M., Forster, C., Schmidhuber, J., Di Caro, G., Scaramuzza, D., and Gambardella, L. M. (2016). A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots. *Robotics and Automation Letters*, 1(2):661–667.
- [Hadsell et al., 2009] Hadsell, R., Sermanet, P., Ben, J., Erkan, A., Scoffier, M., Kavukcuoglu, K., Muller, U., and LeCun, Y. (2009). Learning long-range vision for autonomous off-road driving. *Journal of Field Robotics*, 26(2):120–144.
- [Haklay and Weber, 2008] Haklay, M. and Weber, P. (2008). Openstreetmap: User-generated street maps. *Pervasive Computing, IEEE*, 7(4):12–18.
- [Han et al., 2018] Han, J., Zhang, D., Cheng, G., Liu, N., and Xu, D. (2018). Advanced Deep-Learning Techniques for Salient and Category-Specific Object Detection: A Survey. *IEEE Signal Processing Magazine*, 35(January):84–100.
- [Happold et al., 2006] Happold, M., Ollis, M., and Johnson, N. (2006). Enhancing Supervised Terrain Classification with Predictive Unsupervised Learning. *Robotics Science and Systems Conference*.
- [Häselich et al., 2011] Häselich, M., Arends, M., Lang, D., and Dietrich, P. (2011). Terrain Classification with Markov Random Fields on fused Camera and 3D Laser Range Data. *European Conference on Mobile Robots (ECMR)*, pages 153–158.
- [Hawke et al., 2016] Hawke, J., Guru, C., Tong, C. H., and Posner, I. (2016). Wrong today, right tomorrow: Experience-based classification for robot perception. *Springer Tracts in Advanced Robotics*, 113:173–186.

- [Hayes et al., 2011] Hayes, J. G., Oliveira, R. P. R. D., Vaughan, S., and Egan, M. G. (2011). Simplified Electric Vehicle Power Train Models and Range Estimation. *2011 IEEE Vehicle Power and Propulsion Conference*, pages 1–5.
- [Ho et al., 2013] Ho, K., Peynot, T., and Sukkarieh, S. (2013). Traversability estimation for a planetary rover via experimental kernel learning in a Gaussian process framework. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3475–3482.
- [Ishigami et al., 2011] Ishigami, G., Nagatani, K., and Yoshida, K. (2011). Path planning and evaluation for planetary rovers based on dynamic mobility index. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 601–606. IEEE.
- [Kelly, 2006] Kelly, A. (2006). *Toward Reliable Off Road Autonomous Vehicles Operating in Challenging Environments*, volume 25.
- [Kendall et al., 2015] Kendall, A., Badrinarayanan, V., and Cipolla, R. (2015). Bayesian SegNet: model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. *arXiv:1511.02680v1 [cs.CV]*.
- [Kendall and Gal, 2017] Kendall, A. and Gal, Y. (2017). What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? (Nips).
- [Kersting et al., 2007] Kersting, K., Plagemann, C., Pfaff, P., and Burgard, W. (2007). Most likely heteroscedastic Gaussian process regression. In *Proceedings of the 24th international conference on Machine learning*, pages 393–400. ACM.
- [Kim and Kim, 2005] Kim, C. H. and Kim, B. K. (2005). Energy-saving 3-step velocity control algorithm for battery-powered wheeled mobile robots. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 2375–2380. IEEE.
- [Laible et al., 2013] Laible, S., Niaz Khan, Y., and Zel, A. (2013). Terrain Classification with Markov Random Fields on fused Camera and 3D Laser Range Data. *Mobile Robots (ECMR), 2013 European Conference on*, pages 172–177.
- [Lakshminarayanan et al., 2014] Lakshminarayanan, B., Roy, D. M., and Teh, Y. W. (2014). Mondrian Forests: Efficient Online Random Forests. 27:3140–3148.
- [Le and Smola, 2004] Le, Q. V. and Smola, A. J. (2004). Heteroscedastic Gaussian Process Regression.
- [LeCun et al., 2006] LeCun, Y., Muller, U., Ben, J., Cosatto, E., and Flepp, B. (2006). Off-road obstacle avoidance through end-to-end learning. *Advances in neural information processing systems*, 18:739.
- [Leistner and Bischof, 2009] Leistner, C. and Bischof, H. (2009). On-line Random Forests. pages 1393–1400.
- [Libby and Stentz, 2012] Libby, J. and Stentz, A. J. (2012). Using sound to classify vehicle-terrain interactions in outdoor environments. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 3559–3566.
- [Lowry et al., 2016] Lowry, S., Sunderhauf, N., Newman, P., Leonard, J. J., Cox, D., Corke, P., and Milford, M. J. (2016). Visual Place Recognition: A Survey. *IEEE Transactions on Robotics*, 32(1):1–19.
- [Mactavish et al., 2017] Mactavish, K., Paton, M., and Barfoot, T. D. (2017). Visual Triage : A Bag-of-Words Experience Selector for Long-Term Visual Route Following. pages 2065–2072.
- [Maddern et al., 2014] Maddern, W., Stewart, A., and McManus, C. (2014). Illumination invariant imaging: Applications in robust vision-based localisation, mapping and classification for autonomous vehicles. *Proceedings of the*.
- [Martin and Corke, 2014] Martin, S. and Corke, P. (2014). Long-term exploration & tours for energy constrained robots with online proprioceptive traversability estimation. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 5778–5785. IEEE.
- [Mei et al., 2004] Mei, Y., Lu, Y.-H., Hu, Y. C., and Lee, C. S. G. (2004). Energy-efficient motion planning for mobile robots. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 5, pages 4344–4349. IEEE.

- [Mei et al., 2005] Mei, Y., Lu, Y.-H., Hu, Y. C., and Lee, C. S. G. (2005). A case study of mobile robot’s energy consumption and conservation techniques. In *Advanced Robotics, 2005. ICAR’05. Proceedings., 12th International Conference on*, pages 492–497. IEEE.
- [Mei et al., 2006a] Mei, Y., Lu, Y.-H., Hu, Y. C., and Lee, C. S. G. (2006a). Deployment of mobile robots with energy and timing constraints. *Robotics, IEEE Transactions on*, 22(3):507–522.
- [Mei et al., 2006b] Mei, Y., Lu, Y.-H., Lee, C. S. G., and Hu, Y. C. (2006b). Energy-efficient mobile robot exploration. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 505–511. IEEE.
- [Miguel, 2011] Miguel, L. (2011). Variational Heteroscedastic Gaussian Process Regression.
- [Murphy et al., 2012] Murphy, L., Martin, S., and Corke, P. (2012). Creating and using probabilistic costmaps from vehicle experience. *IEEE International Conference on Intelligent Robots and Systems*, 2012:4689–4694.
- [Newson and Krumm, 2009] Newson, P. and Krumm, J. (2009). Hidden Markov map matching through noise and sparseness. In *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*, pages 336–343. ACM.
- [Ondruska et al., 2015] Ondruska, P., Gurau, C., Marchegiani, L., Tong, C. H., and Posner, I. (2015). Scheduled Perception for Energy-Efficient Path Following. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, WA, USA.
- [Otsu et al., 2016] Otsu, K., Ono, M., Fuchs, T. J., Baldwin, I., and Kubota, T. (2016). Autonomous Terrain Classification With Co- and Self-Training Approach. *IEEE Robotics and Automation Letters*, 1(2):814–819.
- [Paz et al., 2016] Paz, L. M., Pini, P., Hester, G., and Newman, P. (2016). The Path Less Taken : A Fast Variational Approach for Scene Segmentation Used For Closed Loop Control. pages 3620–3626.
- [Richter and Roy, 2017] Richter, C. and Roy, N. (2017). Safe Visual Navigation via Deep Learning and Novelty Detection. *Robotics: Science and Systems XIII*.
- [Ross et al., 2015] Ross, P., English, A., Ball, D., Upcroft, B., and Corke, P. (2015). Finding the ground hidden in the grass: Traversability estimation in vegetation. *Australasian Conference on Robotics and Automation, ACRA*.
- [Scharwachter and Franke, 2015] Scharwachter, T. and Franke, U. (2015). Low-level fusion of color, texture and depth for robust road scene understanding. *IEEE Intelligent Vehicles Symposium, Proceedings*, 2015-Augus(Iv):599–604.
- [Shneier et al., 2009] Shneier, M., Shackelford, W., Hong, T., and Chang, T. (2009). Performance evaluation of a terrain traversability learning algorithm in the DARPA LAGR program.
- [Suger et al., 2016] Suger, B., Steder, B., and Burgard, W. (2016). Terrain-Adaptive Obstacle Detection. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3608–3613.
- [Sun and Reif, 2003] Sun, Z. and Reif, J. (2003). On energy-minimizing paths on terrains for a mobile robot. In *Robotics and Automation, 2003. Proceedings. ICRA’03. IEEE International Conference on*, volume 3, pages 3782–3788. IEEE.
- [Sünderhauf et al., 2015] Sünderhauf, N., Dayoub, F., McMahon, S., Talbot, B., Schulz, R., Corke, P., Wyeth, G., Upcroft, B., and Milford, M. (2015). Place Categorization and Semantic Mapping on a Mobile Robot. *arXiv Preprint*, pages 1–8.
- [Thrun et al., 2006] Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G., Lau, K., Oakley, C., Palatucci, M., Pratt, V., and Pasc (2006). Stanley: The Robot that Won the {DARPA} {Grand} {Challenge}. *Journal of Field Robotics*, 23(9):661–692.
- [Tiwari et al.,] Tiwari, K., Xiao, X., and Chong, N. Y. Estimating Achievable Range of Ground Robots Operating on Single.

- [van der Maaten and Hinton, 2008] van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-SNE. *Journal of machine learning research*, 9(Nov):2579–2605.
- [Vitter, 1985] Vitter, J. S. (1985). Random sampling with a reservoir. *ACM Transactions on Mathematical Software*, 11(1):37–57.
- [Wang et al.,] Wang, G., Irwin, M. J., Berman, P., Fu, H., and Porta, T. L. Optimizing Sensor Movement Planning for Energy. pages 215–220.
- [Weiss et al., 2012] Weiss, M., Helfenstein, A., Mattern, F., and Staake, T. (2012). Leveraging smart meter data to recognize home appliances. In *Pervasive Computing and Communications (PerCom), 2012 IEEE International Conference on*, pages 190–197. IEEE.
- [Yang and Jin, 2006] Yang, L. and Jin, R. (2006). Distance metric learning: A comprehensive survey. *Michigan State University*, 2(2):4.