Learning a Terrain Model for Autonomous Navigation in Rough Terrain

Carl Wellington

CMU-RI-TR-05-59

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Robotics

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

December 2005

Thesis Committee:
Anthony Stentz, Chair
Alonzo Kelly
Jeff Schneider
John Reid, John Deere

©Carl Wellington MMV

This research was sponsored in part by John Deere under contract 476169

Abstract

Current approaches to local rough-terrain navigation are limited by their ability to build a terrain model from sensor data. Available sensors make very indirect measurements of quantities of interest such as the supporting ground surface and the location of obstacles. This is especially true in domains where vegetation may hide the ground surface or partially obscure obstacles.

This thesis presents two related approaches for automatically learning how to use sensor data to build a local terrain model that includes the height of the supporting ground surface and the location of obstacles in challenging rough-terrain environments that include vegetation. The first approach uses an online learning method that directly learns the mapping between sensor data and ground height through experience with the world. The system can be trained by simply driving through representative areas. The second approach includes a terrain model that encodes structure in the world such as ground smoothness, class continuity, and similarity in vegetation height. This structure helps constrain the problem to better handle dense vegetation.

Results from an autonomous tractor show that the mapping from sensor data to a terrain model can be automatically learned, and that exploiting structure in the environment improves ground height estimates in vegetation.

Acknowledgements

First, I would like to thank my advisor Tony Stentz for his insight, excitement, and the freedom he gave me. I'd like to thank committee members Al Kelly and Jeff Schneider for their valuable feedback that kept me on track.

Thanks to John Reid, Zach Bonefas, and the rest of the robotics team at John Deere who have been extremely supportive of this work and have been active partners in the project by providing feedback and expertise in the problem domain, as well as working hard to create a project schedule that allowed real research accomplishments to occur.

I'm especially thankful for the great collaboration with Aaron Courville on the spatial modeling work in this thesis, even when our discussions would more often turn to caribou.

I feel very fortunate to have worked with Cris Dima, the other half of the autonomous tractor project. He has been there from the beginning, and regardless of the task or the hour, I could count on him to carry his share and then some.

Thanks to Drew Bagnell for his insight and for our enlightening discussions, whether at the whiteboard or on the bike trail, and to Sidd Srinivasa for his valuable feedback.

My thanks to Herman Herman for having such a wonderful name and solving all forms of problems on our project, and thanks to all the other people at REC who made our tractor project possible.

I would like to thank Jeff Schneider for the use of his code for locally weighted learning and Stefan Schaal for the use of his code for incremental locally weighted learning.

Thanks to Janet Mather for helping me stay productive in the final stretch and reminding me to keep breathing.

Many thanks to my parents Nancy and David Wellington, for encouraging me and supporting me from the very beginning.

Finally I'd like to thank my wife Jenny for so many things, but especially for understanding and optimism even when the hour is late, and for making me laugh always.

This research was sponsored in part by John Deere under contract 476169

Contents

List of Figures

Chapter 1

Introduction

Automated vehicles that can safely operate in rough terrain hold great promise. They could improve safety by removing people from dangerous environments, they could increase productivity by allowing robots to perform dull and repetitive tasks, they could yield environmental benefits from more precise operation, and they could offer a greater ability to explore difficult or dangerous domains on earth and other planets.

Even if a vehicle is not fully autonomous, there are benefits from having a vehicle that can reason about its environment to keep itself safe. Such systems can be used in safeguarded teleoperation or as an additional safety system for human operated vehicles.

Capable and safe automated vehicles would benefit many applications in agriculture, mining, and the exploration of hazardous areas. Operating in the outdoor environments common in these applications requires a vehicle to recognize untraversable areas and terrain interactions that could cause damage to the vehicle. This is a challenging task because available sensors make very indirect measurements of quantities of interest such as the supporting ground surface and the location of obstacles. Vegetation further complicates the situation by covering and hiding the supporting ground surface, preventing a purely geometric interpretation of the world. In many agricultural applications, the vehicle is required to drive through vegetation, and in more general off-road exploration tasks, driving through vegetated areas may save time or provide the only possible route to a goal.

1.1 Applications

There are a number of applications that would benefit from an automated vehicle that can operate in rough terrain. These applications are mostly characterized by traits that make automation appealing such as repetition, precision, or a dangerous environment. However, these applications are generally outdoors in unstructured or semi-structured areas with many challenges and hazards that prevent factory automation techniques from being successful.

Agricultural tasks are generally repetitive and performed in a known environment that often has structure such as crop rows that can be exploited as navigation aids. The vegetation in agricultural settings usually has a fairly constant height, which can be used to estimate where the ground is even when it's hidden by dense vegetation. Prior maps are often available, and different tasks such as planting and harvesting are performed over the same piece of land so a system can use previously trained paths. Precision agriculture has already brought gps-based mapping and guidance to these domains, and further automation would allow new techniques such as spraying at night, when less pesticides can be used because the bugs are more active and the winds are lower. These domains also present challenges because environmental conditions and the terrain can change, vegetation prevents a purely geometric view of the world, a large variety of obstacles can be present, and people, animals, and other vehicles can enter the workspace unexpectedly.

Surface mining is another repetitive task in a constrained environment. There is often little vegetation and access to these areas is limited, but vehicles need to move fast over changing terrain through different weather conditions, often with other people or vehicles in the area.

Robotic vehicles can perform exploration, search and rescue, and data collection in difficult or hazardous areas on earth and in space. Terrestrial applications include hazardous areas due to nuclear waste, pollution, land mines, war, or extreme environmental conditions. There is often uncertainty in the environment in these cases, and the system may need to handle vegetation and obstacles.

Environments beyond our planet are very hazardous for humans and present a natural application of robotic technology. Current rovers exploring Mars travel very slowly and conservatively, but future missions will require larger rovers that can cover great distances. We do not expect vegetation to be an issue for these missions, but achieving the higher performance desired for future missions may require an adaptive approach because many characteristics of these areas are unknown.

Figure 1.1: Automated tractor test platform (John Deere 6410)
Position sensors: differential GPS, 3-axis fiber optic vertical gyro, Doppler radar ground speed sensor, steering angle encoder, four custom wheel encoders
Terrain sensors: high-resolution stereo pair of digital cameras, infrared camera, two SICK laser range-finders mounted on custom actively-controlled scanning mounts

1.2 Local Rough-Terrain Navigation

Autonomous navigation in rough terrain requires consideration of how the vehicle will interact with upcoming terrain in order to keep the vehicle safe, as well as avoiding obstacles such as people that the vehicle could harm. Although the high-level planning problem is important for a fully autonomous system, this work focuses on the *local* navigation problem, where the goal is to follow a predefined path or head towards a known destination while staying within various safety constraints. This evaluation is limited to the local area near the vehicle, where sensor information about the terrain is available. Depending on the specific application, the vehicle should avoid an unsafe area on the path or stop and call for help. We consider terrain that is common in the applications listed above including slopes, ditches, and vegetation, as well as relevant obstacles such as people, equipment, posts, and buildings.

A typical robot that operates in these types of environments has a set of sensors that give it an estimate of its position and another (perhaps overlapping) set of sensors that measure the terrain in its local environment. Figure 1.1 shows the test vehicle used in this thesis (see appendix A for more details). Most sensors for position estimation are fairly low dimensional and directly produce quantities of interest about the state of the vehicle. A wheel encoder measures forward motion, a gyro measures yaw rate, and gps measures global latitude. The measurements are not perfect, but accurate noise models are known for these sensors, and effective approaches for combining the various sensors such as the Kalman filter [Maybeck, 1979a] have been successfully used in practice for decades.

On the other hand, sensors that measure the environment, such as stereo color cameras, laser range-finders, infrared cameras, and radar produce massive quantities of data, but cannot directly measure many quantities of interest such as the height of the supporting ground surface in vegetation, the friction coefficient of the terrain ahead of the vehicle, or the

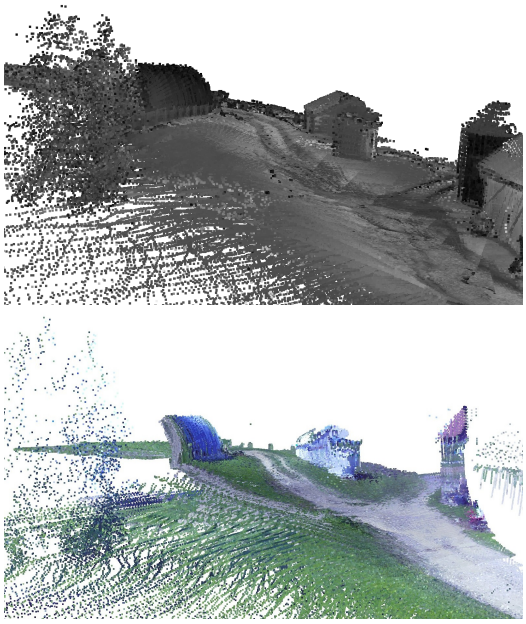


Figure 1.2: Map of farm test site buildings using both ladars as the vehicle drove on a path to the field. The top figure shows laser reflectance, and the bottom figure shows the colorized points (the color camera has a narrower field of view than the laser).

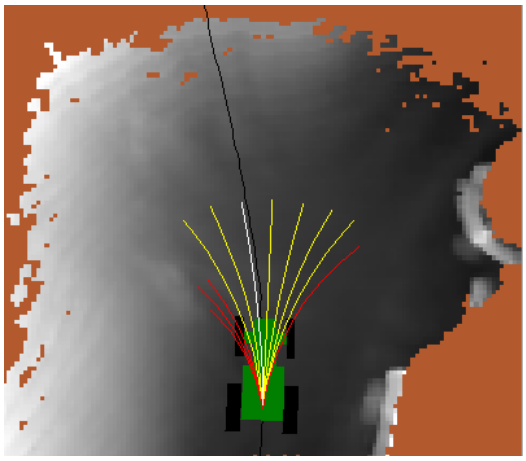


Figure 1.3: Elevation map of farm test site buildings (tractor is in lower right of figure 1.2) using the mean height in each grid cell. Model predictive control showing the chosen arc along the path and unsafe arcs to the left due to the steep slope and to the right due to the farm building.

location of obstacles. This thesis will explore methods of using these indirect measurements to find such quantities of interest.

We assume range sensors such as stereo cameras or the scanned ladars on our test platforms are calibrated and registered with the vehicle pose so sensor data can be accumulated into a high quality global terrain map (see Figure 1.2). We project range measurements back into the camera image planes to tag each range point with color and infrared data. The system maintains a grid representation of the area around the vehicle to hold the data from the forward looking sensors. This approach makes it easy to combine range information from multiple sensors and to combine sensor information over time as the vehicle drives.

We use an approach similar to [Lacaze et al., 2002] to take advantage of the added information about free space that a range measurement provides (see Figure 1.4). Each grid cell contains a column of voxels that record the locations of any hits in that volume of space, as well as the number of rays that pass through that voxel. Maintaining these statistics enables us to compute a rough density estimate to help discriminate between penetrable vegetation and solid obstacles.

To reduce the dimensionality of the data, features can be computed for each grid cell. For example, Figure 1.3 shows an elevation map corresponding to the lower right of Figure 1.2 that uses the average height of all the points in each cell.

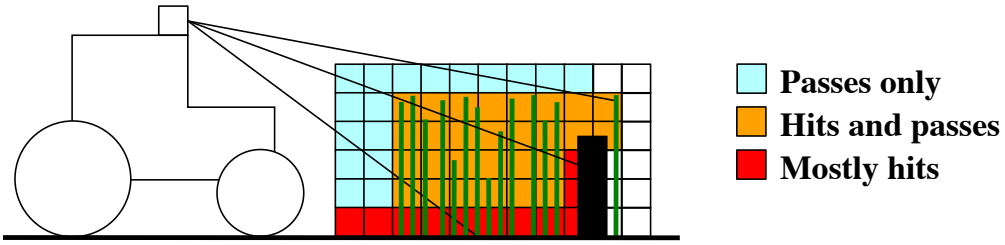


Figure 1.4: A vertical slice of the voxel data structure in a scene with thin vegetation and a solid obstacle. Range measurements are traced through the voxels, which keep track of hits and pass-throughs to separate free-space, vegetation, and ground or other solid substances.

Autonomous navigation in rough terrain has been successfully implemented within a model predictive control framework as a search through speed and steering angle commands over a fixed path length [Kelly and Stentz, 1998b]. In this framework, a vehicle model is used with a terrain model to predict the behavior of the vehicle for different control inputs, and then the predictions are evaluated to choose the best control. Figure 1.3 gives an example of this approach. A set of controls are sampled from the space of possible speed and steering angle commands, and the vehicle and terrain models are used to predict the result of using these controls. The vehicle model includes system delays and the dynamics of the steering actuation to produce feasible arcs.

A kinematic model of the vehicle is then placed on the terrain map at regular intervals along the predicted trajectory, and the heights of the four wheels are found in order to make predictions of the vehicle roll and pitch and ensure that the front rocker suspension is within limits. The heights of the terrain cells under the vehicle are used to check for clearance hazards. The vehicle model can be extended to include various implements so that they are checked for clearance hazards as well.

Once a set of possible arcs are found that satisfy the safety constraints, a cost function is used to evaluate the arcs to find the best control. For path tracking, the cost is the error between the arc and the desired path. If a destination does not have a specific path associated with it, the cost is defined as the minimum distance between the arc and the goal point. By choosing the lowest cost arc that satisfies the safety constraints, the vehicle is able to smoothly avoid obstacles or other dangerous terrain conditions and then re-acquire its path. The system checks speed choices from fastest to slowest, which results in a graceful slowdown as the vehicle approaches an obstacle.

This approach works well if a local terrain map is available that includes the height of the supporting surface. For smooth terrain with solid obstacles, this is straightforward because accurate predictions of the load-bearing surface can be found by simply averaging the height of the range points in the terrain map. However, this performs poorly in vegetation since many laser range points hit various places on the vegetation instead of the ground, as shown in Figure 1.5. Using the lowest point in each grid cell correctly ignores the scattered range points that hit vegetation, but in many cases the lowest laser point does not penetrate denser vegetation, as shown in Figure 1.6. Also, using the lowest point will cause the system to ignore positive obstacles that rise above the ground surface such as a person. Clearly, a more advanced criteria than simply using the average or lowest height in a cell is

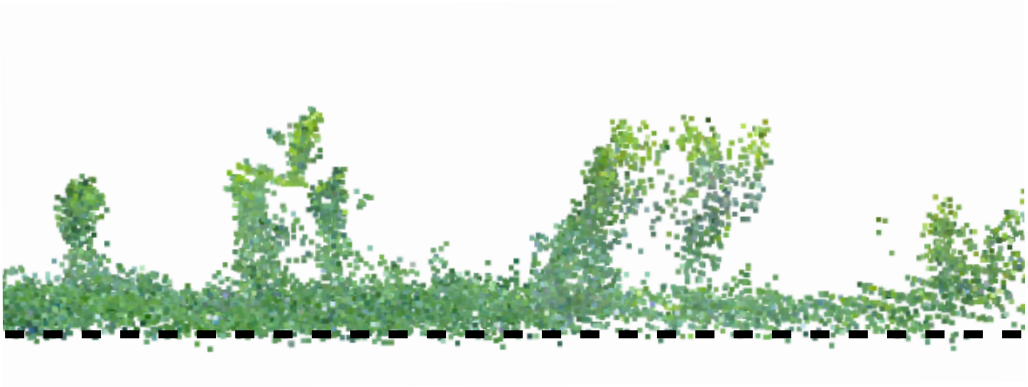


Figure 1.5: Side view of penetrable vegetation data, with approximate ground height. There are range points on the ground and the vegetation.

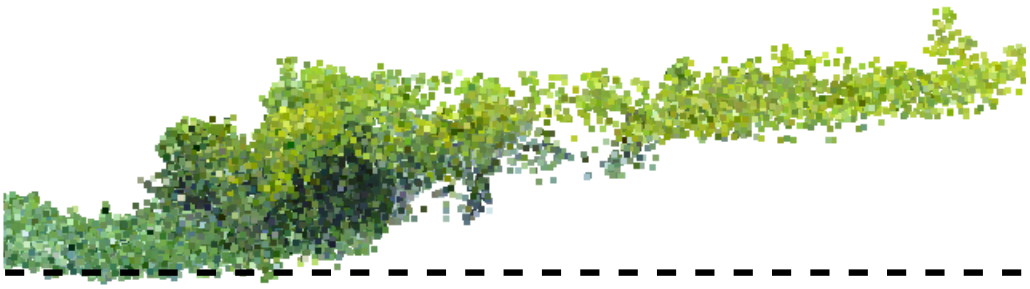


Figure 1.6: Side view of data showing transition from low grass on the left to tall dense non-penetrable vegetation on the right, with approximate ground height. In the tall dense vegetation, there are only range points on the top of the vegetation and the ground remains hidden.

needed to find a useful estimate of the ground plane in vegetation. However, it is difficult to come up with such a rule, and tuning to a given environment can be time consuming and challenging.

1.3 Problem Statement

Current approaches to local rough-terrain navigation are limited by their ability to build a terrain model from sensor data. Available sensors make very indirect measurements of quantities of interest such as the supporting ground surface and the location of obstacles. This is especially true in domains where vegetation may hide the ground surface or partially obscure obstacles.

This thesis investigates methods of building a local terrain map from sensor data that includes the height of the supporting ground surface and the location of obstacles. A

successful system should adapt easily to new conditions, handle incomplete and ambiguous data, find obstacles in vegetation, make predictions of the ground surface even in dense vegetation, and it should run on a real autonomous platform.

1.4 Previous Work

This section summarizes the most relevant work in this problem domain. Chapter 2 presents a more detailed survey of related work in rough-terrain navigation.

As described in section 1.2, local autonomous navigation in outdoor environments is often performed in a model predictive control framework that searches over dynamically feasible control arcs for a safe trajectory [Kelly and Stentz, 1998b]. In this framework, a terrain model with obstacles and the supporting surface is used in combination with a model of the vehicle to find a dynamic trajectory that avoids obstacles while protecting against roll-over, body collisions, high-centering, and other safety conditions. While faithful models of vehicle dynamics are often available, acquiring an accurate terrain model that includes a description of the load-bearing surface and any obstacles in the local environment remains a considerable challenge.

Early work in terrain perception operated in environments with smooth terrain and discrete obstacles [Daily et al., 1988] [Kelly and Stentz, 1998b] and achieved good results in these domains by looking at the height of the sensor readings from individual grid cells. However, as described in section 1.2, the presence of vegetation makes the problem much more difficult because the range points from forward looking sensors such as stereo cameras or a laser range-finder do not generally give the load-bearing surface. Classification of vegetation [Davis et al., 1995] (also common in the remote sensing community - see references in [Hebert et al., 2002]) is not sufficient for this task because a grassy area on a steep slope may be dangerous to drive on whereas the same grass on a flat area could be easily traversable.

A number of researchers have investigated methods that use range data to discriminate sparse vegetation (as shown in Figure 1.5) from solid substances such as the ground or obstacles. These techniques exploit the fact that range measurements often penetrate sparse vegetation, but do not penetrate solid obstacles. The properties used for discrimination fall into two categories: shape and density.

Shape methods begin with a 3D cloud of range points and look at local features of the data points to discriminate between the random spread of points in sparse vegetation and the organized structure of points on solid objects. Researchers have modeled the statistics of laser penetration in grass to find solid objects [Macedo et al., 2000], and they have compared measurements across time and space to filter out areas where the penetration is continually changing [Castano and Matthies, 2003]. A comparison between techniques that look for the range shadow of solid obstacles and techniques based on local point statistics is given in [Hebert et al., 2002]. The idea of computing local statistics about the spread of points was expanded in [Vandapel et al., 2004] to discriminate between sparse vegetation, solid surfaces, linear structures such as branches, and even concertina wire [Vandapel and Hebert, 2004].

Density methods attempt to use range measurements to explicitly measure the density of objects in the environment. This has been done by dividing the world into small

volumes of space and then maintaining density scores (see Figure 1.4) by keeping track of ladar hits and pass-throughs [Lacaze et al., 2002] or ladar and radar measurements [Ollis and Jochem, 2003].

The above methods have shown promising results in sparse vegetation, but they do not address the problem of estimating the ground surface in dense vegetation where the ground is completely hidden as in Figure 1.6. One reason for this is that these methods make the strong assumption of independence between terrain patches and make predictions locally without including spatial context. This can make it difficult to disambiguate data from tall vegetation and data from short vegetation, resulting in poor estimates of the hidden ground height. We hope to address this by relaxing the independence assumption through the inclusion of spatial correlations.

Also, the above methods often contain many rules that can be difficult to construct and hard to tune to a given application. If the parameters are not tuned properly, the system may produce incorrect predictions which could lead to poor vehicle behavior. Some researchers have investigated using labeled training data to learn the parameters automatically [Vandapel et al., 2004], but this approach requires large quantities of hand-labeled training examples which is difficult and time-consuming. One way to improve on this situation is by letting the system learn and adapt automatically using feedback from its environment.

Researchers have investigated the use of parameter identification techniques with soil models to estimate soil parameters on-line from sensor data [Iagnemma et al., 2002b], but these methods only determine the terrain that the vehicle is currently traversing. We would like to make predictions of the terrain in front of the vehicle so that the system can take appropriate action before it reaches these areas.

Other researchers have presented algorithms for color classification that can adapt to slowly changing conditions [Ollis and Stentz, 1997] [Ulrich and Nourbakhsh, 2000] or even new terrain types [Batavia and Singh, 2001] by training online to the colors that the vehicle is currently driving over. The key insight used in all of these methods is that by measuring a quantity of interest in a different way, the system is able to automatically collect training data and improve its predictions. For example, the indoor robot in [Ulrich and Nourbakhsh, 2000] knows that the area it has just traversed is free of obstacles, so it uses that stored data to train a simple color-based obstacle detector. The robot lawn mower in [Batavia and Singh, 2001] assumes that if there is a large area in front of the robot with an unknown color but low height, that area is passable so its colors are added to the training data for a simple color-based obstacle detector. We would like to generalize these adaptive ideas to a greater range of sensor data and apply the approach to the problem of building a terrain model consisting of ground height estimates in vegetation instead of just adapting our color distributions.

In summary, current approaches cannot handle difficult environments that include dense vegetation, which is required for many important applications. Also, current approaches generally have many hand-tuned parameters, which makes them difficult to use or apply to a new application.

1.5 Thesis Statement

This thesis shows that the mapping from sensor data to a terrain model can be automatically learned, even in difficult terrain that includes dense vegetation. Additionally, it argues that including spatial correlations can improve the learned terrain model.

1.6 Overview of Approach

This thesis presents two related approaches for automatically learning how to create a terrain model from sensor data. The first approach uses an online learning method that directly learns the mapping between sensor data and ground height through experience with the world. The system can be trained by simply driving through representative areas. The second approach includes a terrain model that encodes structure in the world such as ground smoothness, class continuity, and similarity in vegetation height. This structure helps constrain the problem to better handle dense vegetation.

1.6.1 Learning a Terrain Model

Vehicle navigation systems that interpret range and appearance data are complex and have many parameters that need to be optimized for good performance. It is difficult and time-consuming to manually tune these parameters, so the idea of a learning system that can easily optimize its own parameters is very appealing. The first part of this thesis investigates how this can be done on an autonomous vehicle operating in rough-terrain.

A critical component of learning any task is receiving appropriate feedback. The type of feedback separates the learning problem into several categories, described in the machine learning community as unsupervised learning, reinforcement learning, and supervised learning. In an unsupervised learning task, the system receives no feedback about desired outputs, so it can only model or cluster the input data. In reinforcement learning, the system is occasionally provided with feedback about how well it is doing, in the form of a reinforcement signal consisting of a reward or punishment, but the system receives no feedback on its outputs so it must determine the connection from input data all the way to the actions that will maximize its reward. In supervised learning, the system is provided with continual feedback in the form of labeled training examples that describe the desired output for a given input. This reduces the learning problem to a function approximation or classification task.

For an autonomous vehicle operating in rough terrain, unsupervised learning is of limited use since the system needs to respond appropriately to different types of terrain, which it can only do if it receives some kind of feedback.

The reinforcement learning problem in this domain may involve letting the vehicle drive around, and then giving it a reward when it reaches a goal and punishing it when it does (or is about to do) something unsafe. This is a very challenging problem, especially when the vehicle is large, expensive, and dangerous, and it will not be considered in this thesis.

Supervised learning is a more tractable problem, but to work well it generally needs large amounts of training data. Acquiring this training data can be expensive and time-consuming, especially if a human is needed to label each training example with the correct output. In this thesis, we use a different approach, and set the problem up in a way that lets

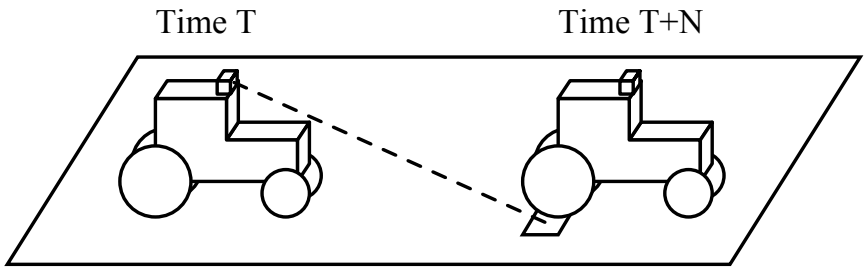


Figure 1.7: Learning a terrain model. At time T, features from forward looking sensors are used to predict the ground height in front of the vehicle. Then, at time T + N the vehicle drives over that area and finds its true height. The system uses this feedback to learn better predictions.

the system acquire labeled training data automatically with minimal human intervention by letting the system receive feedback directly from the environment.

Section 1.2 argued that an accurate local terrain map of the area in front of the vehicle that includes the heights of the supporting ground surface and the location of obstacles would allow successful autonomous navigation. Autonomous vehicles can sense the ground surface in two different ways. First, the system can use forward looking sensors, such as cameras and laser range-finders, which make indirect measurements of the ground, especially if the ground is hidden under vegetation. Second, the vehicle can accurately sense where the ground is underneath its wheels when it drives over an area.

As shown in Figure 1.7, the mapping from forward looking sensors to future vehicle state, which would allow autonomous navigation, can be automatically learned by using the vehicle's experience when it interacts with the terrain. The vehicle uses forward looking sensor data to make predictions of the supporting surface in front of the vehicle, and it stores the sensor data used to make these predictions. Later, when the vehicle drives over that area, it finds the true supporting ground surface under its wheels, and it correlates that desired output with the input sensor data it previously stored for that location. By driving over representative terrain, the vehicle can easily collect massive amounts of labeled training data, which can be used to learn how to construct an accurate terrain map from forward sensor data. This allows the system to train itself to a domain with minimal human oversight. Also, by continuing this learning process while the system is running autonomously, the system can adapt to slowly changing conditions with no human intervention at all.

1.6.2 Using Structure to Constrain Problem

A limitation of the approach presented in the previous section is that predictions are made independently in each small patch of terrain, without including any spatial context. This can cause problems when there is ambiguous feature data. Figure 1.8 shows an example where the range points from a small patch of tall vegetation and a small patch of short grass appear the same. If the system considers each of these patches independently, it will give the same ground estimate for both and get at least one of them wrong. When humans look at the data, it is clear from the context that these are different cases. In the second

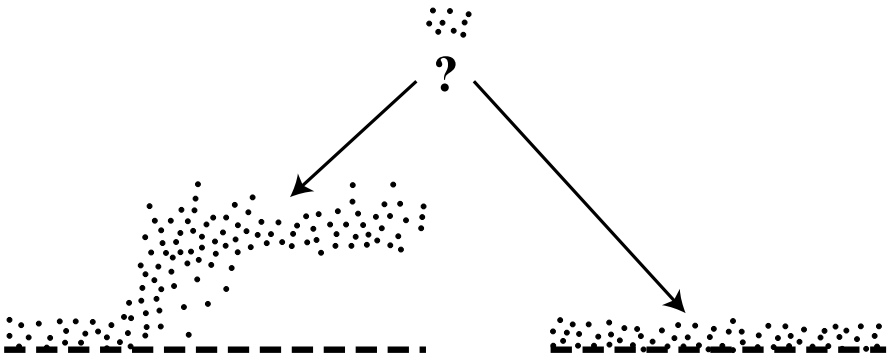


Figure 1.8: Using context, the side view of laser points on the left appears to be a transition from short grass to tall constant height vegetation, and the laser points on the right appear to be short grass on flat ground. These two situations can be ambiguous when looking at the laser points from a single patch without spatial context.

part of this thesis, we relax the strong assumption of independence between terrain patches through the inclusion of spatial correlations. We present a terrain model that can use spatial context to disambiguate similar data features and find the ground height in dense non-penetrable vegetation.

Outdoor environments such as those encountered in agriculture, mining, and the exploration of hazardous environments are often viewed as being "unstructured". However, such environments do possess a great deal of structure that humans frequently exploit in the performance of tasks we wish to automate. For example, consider a vehicle navigating through a field of vegetation or crop. We can use the knowledge that the ground is generally smooth and the vegetation has approximately constant height to infer the ground height and allow navigation even through areas where the ground is not directly observed.

We present a generative, probabilistic approach to modeling terrain that includes three spatial assumptions to help constrain the problem: smooth ground height, class continuity, and similar vegetation height. The structure in the terrain model is combined with information from multiple sensors on the vehicle using sensor models that are automatically learned from training data by driving over representative terrain, as in the first approach described in the previous section. We treat obstacles as having uncertain appearance so that we do not need to train explicitly on obstacles. We rely on accurately modeling the known classes and detect obstacles when no other class in the model is consistent.

Joint inference of ground height, class height and class identity over the whole model can result in more accurate estimation of each quantity. Inferring the vegetation height gives an improved estimate of the height of the underlying ground. Knowing the ground height helps disambiguate solid obstacles standing on top of the ground from the solid ground surface. As an example of these ideas, Figure 1.9 shows a challenging scene with a person wearing a camouflage jacket standing in tall dense vegetation and Figure 1.10 gives the terrain model estimate that was inferred from the color and infrared tagged range data using our model structure with trained parameters. Figure 1.10 shows the inferred ground surface, an obstacle where the person is standing, and classification results for the dirt

Figure 1.9: View from tractor of person, tall weeds, low grass, and small dirt mound

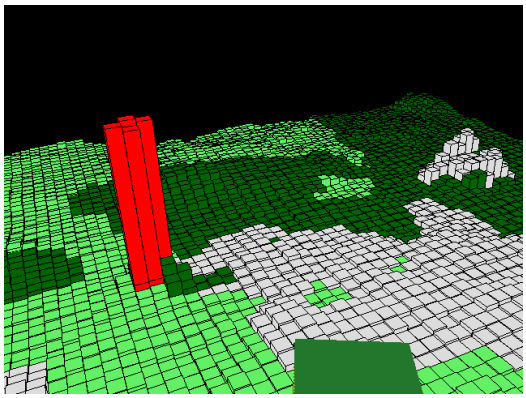


Figure 1.10: Model estimate showing obstacle location, ground heights, and terrain classifications (from light to dark: dirt, tall weeds, low grass)

mound, the tall yellow weeds, and the low grass. The system leverages spatial structure in the environment to find the person, make accurate ground height predictions even though the ground is hidden under dense vegetation, and produce classification results that are generally correct (except for the large patch of dirt in front of the dirt mound, but that area of the ground is hidden behind the tall weeds in front of the tractor). This example is discussed in more detail in section 6.8.4, and we present results in chapter 7 that show that including spatial correlations produces improved ground height estimates in vegetation.

1.7 Organization

The remainder of this thesis describes the above approaches in more detail and presents results from realistic test settings in challenging areas that include various obstacles, hazards, and vegetation. Chapter 2 describes related work in rough-terrain navigation in more detail. Chapter 3 gives the general online learning approach, and chapter 4 shows how this approach can be applied to the problem of learning a terrain model in vegetation. Chapter 4 also includes results and discusses the limitations of this approach. Chapter 5 gives background information on various forms of probabilistic Markov models, which are then used in chapter 6 to incorporate structure into the terrain model. Chapter 6 includes results using this terrain model in vegetation, and chapter 7 gives a comparison between the two approaches. Chapter 8 presents conclusions and areas for future work. System details are given in Appendix A.

Chapter 2

Related Work in Rough-Terrain Navigation

A summary of closely related work was given in section 1.4 as motivation for the approaches presented in this thesis. This chapter gives a more general survey of related work in rough-terrain navigation.

Rough-terrain navigation generally falls into two different types of problems: global planning and local navigation. The global planning problem requires the vehicle to navigate to a goal destination in an unknown or partially known environment. This scenario is common in exploration problems and requires higher level planning to find a path that satisfies safety constraints and is globally optimal, or at least feasible. Because of the high dimensionality of the rough terrain planning problem, researchers generally use a high level algorithm to find promising paths or subgoals quickly, and then more accurate models are used to evaluate and perhaps modify the local trajectories to find a feasible path. Although we are not investigating the global planning problem in this work, much of the prior work in rough terrain navigation approached the problem from a planning perspective, and their techniques for the evaluation of local paths are directly applicable to our proposed work.

The second type of problem involves local navigation. In this case, a vehicle may have a fixed path that it is attempting to follow, or a direction that it is trying to move towards. This occurs in many agricultural settings where the path is predetermined by the crop and the vehicle has very limited maneuverability to deviate from the path. In this case, the safety problem becomes one of determining controls that keep the vehicle close to the path and within certain safety limits. Other domains are less constrained, and so various methods of arbitrating between different objectives are used. As in the planning problem, various methods are used to evaluate potential control actions for safety and performance.

The following sections describe prior work in areas relevant to rough terrain navigation, including global planning, local navigation, agriculture, and automatically learning terrain parameters.

2.1 Global Planning

Although the focus of this thesis is on local navigation using real data, much of the prior work in rough-terrain navigation has been in the context of planning. The following research

was done in simulation, but includes a number of safety constraints based on the vehicle or terrain.

[Shiller and Gwo, 1991] uses a fast global planner to find a number of candidate paths through a hilly terrain that includes discrete obstacles and low-traction areas. A simple dynamic model is then used to find the velocity limit curve along the path that satisfies constraints on engine torque, sliding, contact, and tip-over, and the path with the shortest overall time is chosen. This technique was then extended [Shiller, 2000] to allow traversal over small obstacles by using a more general traversability metric instead of binary obstacles. Simulation results are given and complete terrain knowledge is assumed.

[Cherif and Laugier, 1995] [Cherif, 1999a] [Cherif, 1999b] also use a high level planner to find subgoals, and then uses complex models of deformable terrain, robot dynamics, and the wheel/ground interaction to find a feasible trajectory between the subgoals. Constraints on collision, contact, tip-over, sliding, torque, and dynamic feasibility are included, but perfect knowledge of the terrain and robot are assumed and the planner is slow. Results are given in simulation for a six-wheeled rover traversing areas that include static obstacles, slippery areas, and hilly terrain.

[Chanclou and Luciani, 1996a] [Chanclou and Luciani, 1996b] use a potential field based global planner along with complex models of the vehicle and terrain for local planning. The models are based on point masses and connections, and can simulate complex interactions between the vehicle and the soil, including wheel slip and soil deformation. Simulation results are given showing realistic wheel-soil interactions, but perfect knowledge is assumed and the simulation is slow.

[Amar et al., 1993] [Amar and Bidaud, 1995b] [Amar and Bidaud, 1995a] use a very simple planning scheme based on adjusting the parameters of a cubic spline path between known subgoals to minimize a cost function on the path. A kinematic model is constructed to evaluate the paths. The model includes wheel-soil interactions and constraints on stability, slipping, torque, and joint-angle. Simulation results are given.

2.2 Local Rough-Terrain Navigation

The following sections describe efforts by different groups at creating vehicles that can successfully navigate in rough terrain. Research is organized in roughly chronological order within each group.

2.2.1 Hughes

A team from Hughes Artificial Intelligence Center reported the first autonomous operation of a robotic vehicle in natural terrain in the late 80's [Daily et al., 1988]. Although the vehicle's speed was limited by the computational resources available at the time, their vehicle was able to successfully navigate on natural terrain, and their basic approach is very similar to the methods being used today. A hierarchical control system is used to combine a map based planner with low-level behaviors. One of the behaviors is responsible for evaluating safe trajectories. The points from a 3D laser range finder are inserted into an elevation grid map. A vehicle model is placed onto the elevation map at points along the trajectory, and tests are performed for high slope, clearance, and suspension limits to determine if that trajectory is safe.

2.2.2 CMU

Carnegie Mellon University has a long history of rough-terrain navigation research. The FastNav system [Singh and Keller, 1991] could perform high speed obstacle detection on simple terrain. It fits a function to range points to determine the terrain surface, and then looks for vertical deviations from that surface. The path is known, so only the area around the future path is evaluated, and computational tricks are used to make the system run fast. Using this system, the NavLab vehicle was able to drive at 14 km/hr (9 mph) on rolling terrain with discrete obstacles. This system has since been applied using a radar sensor to a large Caterpillar mining truck that travels up to 40 km/hour (25 mph) on dirt roads.

Early experiments by [Langer et al., 1994] involve discretizing the world and computing a binary traversability measure by testing the laser points in each cell for large height variation, high slope, or a vertical discontinuity. This traversability measure is computed recursively by the SMARTY point-based range processing system as each range point and uncertainty is inserted into the cell [Hebert, 1997]. Using the Distributed Architecture for Mobile Computing [Rosenblatt, 1997], different behaviors (avoid obstacle, seek goal, etc) then vote for steering directions, and a linear combination of votes produces the next steering direction. The system was able to drive the NAVLAB II HMMWV at 7 km/hr (4 mph) over gentle terrain that also included rocks, hills, and ditches. The incremental global path planner D* [Stentz, 1994] was incorporated into the system [Stentz and Hebert, 1995], allowing the vehicle to perform an autonomous 1.4 km traverse in an open area with sparse obstacles.

Instead of classifying the terrain as obstacles or free, [Brummit et al., 1992] maps points from a laser scanner into a grid and then finds local plans using a "guess and check" process that simulates a vehicle model on the terrain and checks for body collisions and static stability. This system was able to drive the NAVLAB II HMMWV at an average speed of 7 km/hr (4 mph) over 4.5 km of gently rolling terrain with sparse obstacles. [Kelly and Stentz, 1998a] [Kelly and Stentz, 1998b] formalizes this idea in terms of predictive control and evaluates a set of dynamically feasible trajectories for hazards including tip-over, body collision, height discontinuity, and unknown terrain. Votes for different control actions are combined by selecting the steering command that best achieves the goal (path following, goal seeking etc) while staying within safety constraints. Through intelligent use of perception data and by accounting for processing delays, the RANGER system was able to control the NAVLAB II HMMWV on excursions of 15 km at speeds up to 15 km/hr (9 mph).

The RANGER system has been quite successful for large vehicles like the HMMWV, but smaller rovers have had difficulty using it in practice because small errors in the elevation map translate to predictions of large tilting hazards or body collisions for a small rover. [R. Simmons and Whelan, 1996] modified Kelly's RANGER algorithm to better work for small rovers by using traversability maps instead of terrain height maps. Planes are fit to terrain range points and a traversability measure is computed from the plane roll/pitch, variation from the plane, and a confidence measure based on the number of points used. Control uncertainty is handled by averaging over a Gaussian distribution of steering angles. The resulting MORPHIN system (a "power" RANGER) was implemented on a rover and performed a 1km traverse 90% autonomously. Merging of traversability maps and the addition of Stentz's D* algorithm for global planning is described by [Singh et al., 2000]. The MORPHIN system was also used on the CMU Hyperion rover

during tests of sun-synchronous navigation [Urmson et al., 2002] and was modified for the 2003 Mars rovers [Goldberg et al., 2002].

More recently, the RANGER system has been combined with Stentz's D* planning algorithm for the PerceptOR project. This system uses a small helicopter that flies in front of the vehicle to provide more sensor information [Stentz et al., 2002b]. The vehicle performs a vegetation penetrability analysis similar to [Lacaze et al., 2002], and has been tested extensively in realistic cluttered rough-terrain environments at speeds of 3 km/hour (2 mph). However, this system required significant manual tuning to work in a given environment. More recent work has included an application of the learning approach given in chapter 3 of this thesis to automatically learn the ground height in vegetation [Kelly, 2004].

[Urmson, 2004] described methods for high-speed navigation on the Sandstorm vehicle by incorporating prior map data and swerving around obstacles instead of stopping for them.

Early work in vegetation focused on classification of vegetation and solid substances [Davis et al., 1995]. Vegetation classification and removal using local statistics of range data points has been investigated in [Hebert et al., 2002]. This work was then extended by [Vandapel et al., 2004] to discriminate between sparse vegetation, solid surfaces, linear structures such as branches, and even concertina wire [Vandapel and Hebert, 2004].

Dima and Wellington have worked on applying machine learning techniques to various portions of the rough-terrain navigation problem within the context of an agricultural application. [Dima et al., 2004b] shows the benefit of fusing multiple classifiers that use different sensing modalities such as color, texture, and ladar to detect obstacles in outdoor environments. Training these classifiers requires large amounts of labeled training data. [Dima et al., 2004a] looked at active learning methods to automatically select a subset of "interesting" images from long data collection runs to achieve good classification results with small amounts of labeling.

The two learning approaches in this thesis are based on earlier publications. The online learning approach using independent cells described in chapters 3 and 4 is based on [Wellington and Stentz, 2003] and [Wellington and Stentz, 2004]. The approach that includes spatial correlations described in chapter 6 is based on [Wellington et al., 2005].

2.2.3 JPL

Various groups at the NASA Jet Propulsion Laboratory have worked on rough-terrain navigation for mars rovers as well as terrestrial vehicles. One group uses reactionary fuzzy control for off-road navigation. Seraji [Seraji et al., 2001] [Seraji and Bon, 2002] [Seraji and Howard, 2002] extracts measures of roughness, slope, and discontinuity from stereo images to determine the traversability for three regions in front of the rover. Simple reactionary fuzzy rules are then used to navigate around obstacles and toward a goal. Fuzzy rules are also used for rover safety. [Tunstel et al., 2001] classifies the upcoming terrain from camera images into three traction classes using a neural network, and then applies fuzzy heuristics to determine the safe speed for the rover based on traction and attitude. Results with a Pioneer robot in natural terrains show successful navigation around obstacles and speed reduction based on changes in traction and attitude.

Early rover technology for Mars mainly involved the slow, tedious process of teleoperation, but current and future rover missions have increased autonomy for higher science

return. [Goldberg et al., 2002] describes GESTALT, which is a port of the CMU MORPHIN software to their Mars rover.

JPL has also performed research for terrestrial off-road navigation. Early work included implementing the RANGER system on a HMMWV [Matthies et al., 1995] using stereo and infrared classification of vegetation. It was able to travel at 5-10 km/hour in smooth terrain with some low vegetation.

The JPL vision group has proposed many perception systems for classifying terrain from forward looking sensors [Bellutta et al., 2000] [Manduchi et al., 2001] [Talukder et al., 2002b] [Manduchi et al., 2005]. Matthies has also proposed specialized techniques for finding water [Matthies et al., 2003a] and negative obstacle [Matthies and Rankin, 2003] hazards.

Using a simple 1D dynamic vehicle model and spring-based models of different classes of terrain, [Talukder et al., 2002a] finds the maximum speed the vehicle can travel without exceeding a vertical acceleration limit. This technique allows the vehicle to drive over a small bush or clump of grass, but avoid a similar sized rock. Talukder also mentions the possibility of learning the spring models online.

[Macedo et al., 2000] finds hard obstacles hidden in grass by assuming a statistical distribution of grass and looking for areas that do not follow that distribution. This analysis was taken further in [Matthies et al., 2003b] by carefully studying how far ladar sensors penetrate various types of vegetation, and including preliminary results using radar.

[Castano and Matthies, 2003] uses a set of time and space locality constraints to filter out sparse vegetation and detect solid obstacles. This system allowed a small robot to navigate through vegetation and avoid a person.

2.2.4 NIST

[Lacaze et al., 1998] described a method of precomputing a number of dynamic trajectories which are then checked against obstacles at run time. The implementation of this system is given in [Coombs et al., 2000]. The system checks the range points from a 3D laser scanner for height discontinuities to create a simple binary obstacle representation in a local grid map. A large set of dynamically feasible trajectories are then tested for intersections with these obstacle cells, and a path is chosen that maintains a speed-dependent safety clearance around the vehicle. For simple rolling terrain with discrete obstacles, their vehicle was able to travel at speeds of 35 km/hr (22 mph).

These ideas have been extended to better handle difficult terrain through the military Demo III program [Lacaze et al., 2002]. The system uses the 4D/RCS reference model control architecture [Albus, 2000] to plan at different time and space resolutions. To handle vegetation, the volume in front of the vehicle is broken up into 3D voxels, which each record the number of laser hits in that cell and the number of times a laser passes through that cell. The ground surface is then found as the boundary between cells that have mostly hits and cells that have mostly pass-throughs. They report that errors rarely exceed 0.25m in tall grass. Planning is performed as a graph search over clothoid arcs that have costs based on path length, side accelerations, density of vegetation, vehicle roll/pitch, roughness, body-collision, and unknown cells. Their system has been demonstrated in difficult realistic conditions with sparse vegetation.

2.2.5 GDRS

General Dynamics Robotics Systems have built several automated vehicle platforms for military applications such as the XUV [Glo, 2005]. This platform has been used for many demonstrations using software jointly developed by NIST [Lacaze et al., 2002] and more recently CMU [Vandapel et al., 2004]. Although there are few publications on this platform or the work done at GDRS, their systems are very capable and have been successfully tested in many realistic tests in challenging terrain with over 400 km logged in combined autonomous and teleoperation tasks [Glo, 2005].

2.2.6 LAAS

The LAAS group has done work in rough-terrain planning. [Simeon and Dacre-Wright, 1993] created a planner that expands a tree of feasible trajectories that satisfy constraints for contact, suspension limits, tip-over, and body collision by statically placing the robot on the terrain. As part of the EDEN experiments at LAAS, this planner was implemented on the ADAM rover in a system that switched between different planning modes depending on the difficulty of the terrain [Lacroix et al., 1994].

[Nashashibi et al., 1994] describes the construction of terrain maps using points from a 3D LADAR and their associated uncertainty. These maps were then used for localization and as an input to the planning system, allowing ADAM to slowly traverse a 50m section of rugged terrain. This planning method was later extended by Hait to explicitly handle sensor and control uncertainty [Hait and Simeon, 1996] and landmark visibility for localization [Hait et al., 1999], but only simulation results are given for these extensions.

2.2.7 MIT

Researchers at the MIT Field and Space Laboratory have investigated rough-terrain planning for space rovers and various ways to estimate important terrain parameters online. [Farritor et al., 1998] employs genetic algorithms to search through a list of discrete rover actions, using a cost function that includes constraints on power consumption, actuator saturation, wheel slip, and vehicle stability. [Iagnemma et al., 1999] instead uses A* to quickly find candidate paths through the terrain. A similar kinematic analysis is used to evaluate the paths, but simple notions of sensor and control uncertainty are included. Simulation results show the importance of the kinematics evaluation.

In more recent work, Iagnemma notes the importance of wheel-terrain interaction on rough-terrain mobility and describes an on-line approach to estimate terrain parameters [Iagnemma et al., 2002b] and wheel-terrain contact angles [Iagnemma et al., 2001]. He also mentions the possibility of associating visual terrain classification with terrain parameter estimation for improved control and planning [Iagnemma and Dubowsky, 2002]. Laboratory experiments using a smooth wheel in sand verify his approach. Iagnemma is also investigating [Iagnemma et al., 2002a] the use of complex dynamic models for high speed rough-terrain navigation.

2.3 Agriculture

A number of researchers have specifically looked at agricultural applications. Many agricultural settings include significant structure such as crop rows that can be used for guidance. [Reid and Searcy, 1987] presented an early vision algorithm that could segment cotton rows and determine heading and offset errors. [Billingsley and Schoenfisch, 1995] demonstrated a similar approach to track straight rows of cotton at up to 25 km/hour. [Gerrish et al., 1997] reported visual guidance in straight row crops at 13 km/hour. [Southall et al., 1999] used known planting geometry to detect individual plants for guidance.

Instead of using local visual features, researchers have developed automatic tractor steering systems using only differential GPS [O'Conner et al., 1996]. [Zhang et al., 1999] developed a system that fused vision-based crop row detection with GPS to provide more robust vehicle guidance.

[Ollis and Stentz, 1997] at CMU built an adaptive classifier to track the cut/uncut line in an alfalfa field for the vision-based guidance of a harvester. This system was able to autonomously harvest hundreds of acres of crop in various fields and lighting conditions [Pilarski et al., 2002]. It also included vision-based techniques for end-of-row detection and simple color-based obstacle detection.

Our early work included a demonstration of GPS-based guidance of a tractor in an orange grove with simple obstacle detection [Stentz et al., 2002a]. Using a pre-taught path, it drove autonomously for 7km at speeds ranging from 5-8 km/hour while pulling a sprayer.

Many researchers have investigated the automatic guidance of farm equipment (see [Reid et al., 2000] for a summary), and this technology has matured to the point where several products are available for automatic steering along straight rows using GPS when an operator is present in the cab. However, very little vehicle safeguarding work has been done in the agricultural domain, which is required for full automation.

2.4 Learning Terrain Parameters

2.4.1 Soil Parameters

[Le et al., 1997], [Iagnemma et al., 2002b] and [Yoshida and Hamano, 2002] use soil models to estimate soil parameters online from sensor data. These methods use wheel slip to determine the traction of the terrain that the vehicle is currently traveling over.

While detecting the current soil parameters is useful, it may not be enough for an autonomous vehicle. The system needs to be able to predict the response to the upcoming terrain to be able to make the correct decision now. For terrain that is uniform or slowly and continuously changing, these methods may work well, but for terrain that has abrupt changes, such as driving off of a gravel road onto muddy soil, these methods may react too late.

2.4.2 Color Classification

A number of researchers have used online adaptation to better handle changes in lighting for color-based obstacle detection.

[Ollis and Stentz, 1997] computes a linear separator to classify cut and uncut crop from color images. The pixels in the current image from each class are used to train the classifier for the next image, which results in adaptive behavior similar to an online EM algorithm. This algorithm was successful at tracking slow changes in lighting and crop variation.

[Ulrich and Nourbakhsh, 2000] stores a histogram of the colors for the area the robot recently drove over and then classifies anything that doesn't match that histogram as obstacles. This allows the system to adapt to slowly changing conditions.

[Batavia and Singh, 2001] similarly stores a histogram of the colors in front of the robot, but adds a height threshold to check if an unknown color is truly an obstacle. If there is a large area in front of the robot that is an unfamiliar color but has low height, then that area is assumed to be passable and those colors are added to the training data.

These approaches have demonstrated the benefits of adapting online, and our work in chapters 3 and 4 builds on these ideas.

2.5 Summary

Current approaches to rough-terrain navigation work well in smooth terrain with discrete obstacles, and have shown promise in more cluttered environments that include vegetation, but they are generally not adaptable and don't handle dense non-penetrable vegetation.

Chapter 3

Online Learning

To overcome the difficulties associated with creating terrain models for a complex environment that may be unknown or changing, we close the loop around vehicle predictions as shown in Figure 3.1 by making predictions from sensor data and then observing actual behavior when the vehicle drives over that area. This feedback is used for continual learning and adaptation to current conditions.

This chapter describes the basic online learning approach, how it fits into vehicle control, the assumptions it makes, and finally discusses some challenges to learning online. Chapter 4 describes the application of this general approach to the problem of finding the supporting ground surface in vegetation.

3.1 Approach

There are a number of terrain characteristics that can be sensed by an autonomous vehicle in multiple ways. For example, the ground surface height can be measured using forward looking range sensors as well as by using the known wheel locations when the vehicle drives over that area. The friction coefficients of the soil could perhaps be inferred from color and infrared camera data and can also be measured by monitoring wheel slip as the vehicle drives over that area. In both of these cases, we would like to be able to make predictions of these parameters for areas ahead of the vehicle, but we expect much higher quality measurements of the parameters under the vehicle. As shown in Figure 3.1, we can correlate the indirect measurements ahead of the vehicle, which we call *terrain features*, with the direct measurements of *terrain parameters* found when the vehicle drives over that area. This data forms input-output pairs between terrain features (e.g. range data points) and terrain parameters (e.g. ground height) that can be used to learn this mapping automatically.

To make this concrete, let us interpret Figure 3.1 for the problem of finding the ground height in vegetation. The system makes predictions of the location of the ground height terrain parameters from terrain features extracted from data from forward looking sensors like ladar or cameras. Whenever the system extracts features for predictions, it stores these features in that cell. Features may change depending on how far away the sensors are, so this process is repeated at different distances to build up a set of feature sets for the cell. Then the vehicle drives over the terrain and measures the true surface height with its rear wheels.

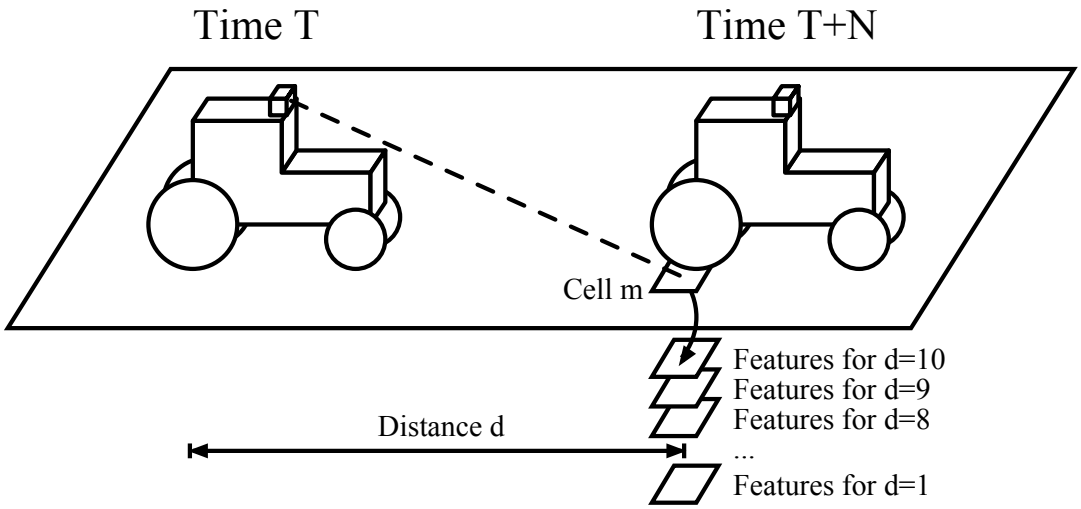


Figure 3.1: Learning terrain height predictions. At time T, features from map cell m at distance d are used to make a prediction and then stored. As the vehicle gets closer to the cell, feature sets for the cell computed at those distances are stored. Then, at time T + N the vehicle traverses the cell and finds its true height. The learner is trained with the set of feature sets computed at different distances and the true height found from the wheel.

All the stored feature sets are correlated with the true height and these input-output pairs are used as training examples to an online learning algorithm (see section 4.3) that learns the mapping from terrain features to the ground surface height terrain parameter. This process happens continuously, so the more the vehicle interacts with the environment, the more training data the learning system receives.

Features for obstacles such as buildings and other vehicles cannot be found by driving over them, so they need to be entered manually. For these untraversable areas, feature sets are stored for different distances just like other areas, but we must manually select them in our user interface to assign a truth value such as the highest laser point for learning (see section 4.1).

3.2 Adaptive Control

This section describes how the online learning approach fits into the vehicle navigation system. We cast the local navigation problem in the optimal control formalism of *adaptive model predictive control* [Sanchéz and Rodellar, 1996]. Instead of reactionary control techniques such as PID control, optimal control seeks to find the set of controls that will optimize some cost function. Model predictive control uses a model to predict what will happen after the application of certain controls. These predictions are then used to find the control that is optimal according to some cost function (as described in section 1.2). Adaptive model predictive control works the same way, but it also uses the experience

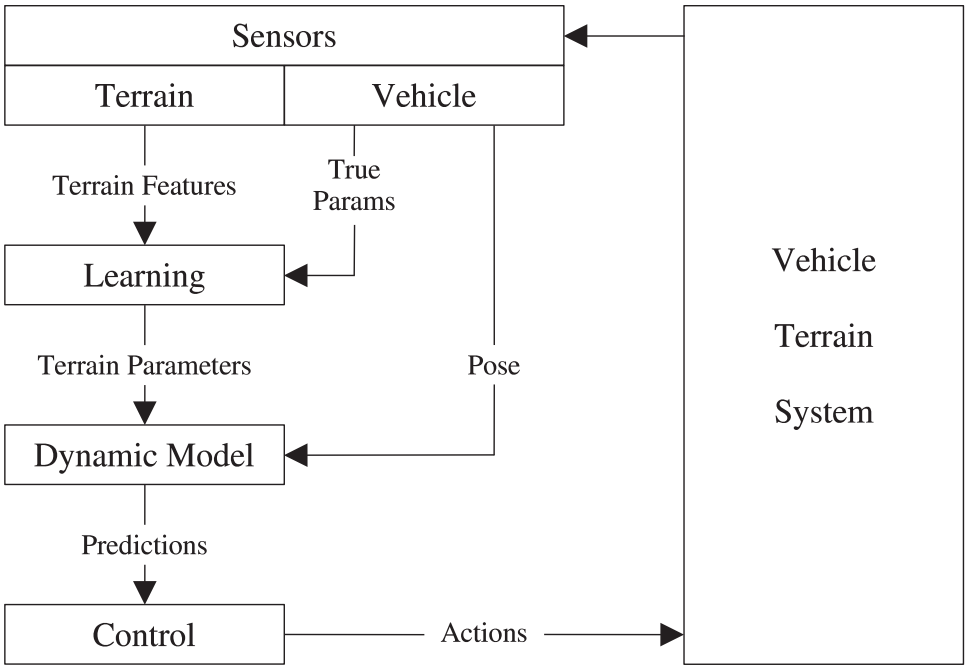


Figure 3.2: Block diagram of proposed approach

from interacting with the world to change its model to make more accurate predictions and therefore produce better control actions.

Figure 3.2 gives a block diagram of the approach. Sensor information about the terrain is transformed into a set of features such as the average height of the laser points or the color of the terrain. These features are presented to an online function approximator that produces estimates of important terrain parameters such as ground height or the friction coefficient. A dynamic model uses this information to make predictions of future vehicle motion, including safety variables such as roll, pitch, and clearance, for a set of possible control actions. The controller then uses these predictions to find the best action that stays within allowable safety constraints. When the vehicle drives over a section of terrain, it measures the actual terrain parameters through its interaction with the terrain, and this information is used to improve the function approximator so that it can better learn the mapping between terrain features from the sensors and terrain parameters for the dynamic model, resulting in more accurate predictions and better control.

This combination of kinematic/dynamic vehicle model equations with learning techniques offers several advantages. Known kinematic relationships do not need to be learned, so the learner can focus on the difficult unknown relationships. Also, the learned function can be trained on flat safe areas, but is valid on steep dangerous areas. If we learned the roll and pitch directly, we would need to provide training examples in dangerous areas to get valid predictions there.

3.3 Assumptions

The central idea of this approach is that learning a mapping from terrain features to important model parameters such as ground height (or other parameters such as surface friction) will allow more accurate vehicle predictions to be made. This brings up three key assumptions.

3.3.1 Accurate Local Position Estimation

It is important that the vehicle knows where it is in the world. For many applications, differential GPS can be used to provide global position to centimeter accuracy. This information can be combined with other on-board sensors (see section A.3) to provide the needed position accuracy. There are a number of other domains (underwater, underground, under dense foliage, other planets) where GPS is not available. Many researchers are actively pursuing solutions to the simultaneous localization and mapping (SLAM) problem [Majumder et al., 2003]. Hopefully work in this area will produce solutions able to provide accurate position estimation in the absence of GPS. Other solutions are also available. For example, in the agricultural domain visual information from the crop rows has been combined with on-board sensors to get a better position estimate [Zhang et al., 1999]. Finally, because this work involves the *local* navigation problem, good relative pose information is enough. The pose estimate must be stable enough that the information from the forward looking sensors can be correctly registered to the behavior of the vehicle when it drives over that area, as shown in Figure 3.1.

3.3.2 Terrain Feature Observability

This approach assumes that qualities of the terrain that will affect the vehicle are observable by available sensors. However, some hazardous situations for a rough-terrain vehicle are undetectable even by humans. A common example is an object or hole completely hidden by uniform weeds, as shown in Figure 3.3. For situations like this, algorithmic solutions won't help, and a better sensor such as vegetation-penetrating radar is needed. This approach can take advantage of such a sensor if it is available, but explicit investigations into advanced sensors are beyond the scope of this work.

3.3.3 Terrain Parameter Observability

The third assumption is that the vehicle will be able to measure the important model parameters by interacting with the terrain. Ground height can be measured using the wheels, and terrain friction can be measured from the wheel slip, but the actual "danger" of a patch of terrain is not directly observable. This approach only considers quantities that are measurable. Even for parameters that are observable such as ground height, we may not be able to recover exactly the true surface. As shown in Figure 3.4, the "truth" values found from the wheel measurements are actually the convolution of the wheel with the ground, so it cannot completely resolve holes or step hazards. This is not a problem in most domains where the ground is relatively smooth.

Classification of obstacles is important for a better understanding of the terrain and possible risks. For example, although a vehicle may react similarly to driving over a person

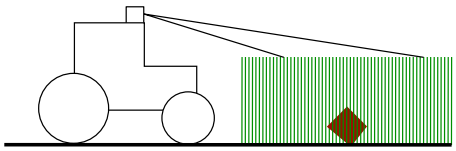


Figure 3.3: Undetectable rock hidden in vegetation

Figure 3.4: Partially unobservable true ground height due to wheel size

lying down and to driving over a similarly sized mattress, clearly the situations require different actions in a fully automated vehicle. Obstacle classification is a very challenging problem and many people are actively researching it. This approach can naturally accept the outputs of an obstacle classification system as inputs as long as the classification labels can be projected into the world frame. Also, the second approach in this thesis presented in chapter 6 will explicitly include obstacle classification.

3.4 Challenges of Online learning

3.4.1 Affecting the Input Distribution

Learning systems generally make the assumption that the training data and the test data are drawn from the same underlying distribution. One concern with any online learning system is that its actions will affect the distribution of input data, and may shift the input distribution to a part of the input space that has not been trained and perhaps cause the system to perform poorly. For example, if a vision system is trained to follow a road by being shown many examples of road images, it may do well when it is on the road, but if it makes a small mistake, then the next input image will have less road in it and therefore be in a part of the input space that is less well known by the system, again increasing the possibility of a mistake. This problem was recognized in the ALVINN neural network driving system [Pomerleau, 1995]. In that work, the problem was addressed by warping the input images to simulate other viewing angles, and thus maintain training data over a larger portion of the input space. They also attempted to maintain a database of representative training examples over everything they've seen to prevent the neural network from overweighting recent training data and forgetting earlier training data.

We don't have these problems because of how we set up the learning problem and our choice of function approximator. Instead of directly learning a cost or driving direction, this approach learns terrain parameters such as ground height that are used with a vehicle model to determine safety. This means that the training data from vegetation in flat safe areas is directly applicable to dangerous slopes that are covered in similar vegetation.

Untraversable areas that do not share common features with traversable areas (e.g buildings, people) require time-consuming manual labeling, so we expect much more training data to be available for traversable areas than these "obstacle" regions. The learning algorithm used in this work (see section 4.3) produces confidence intervals on its output which are based on how much data it has seen in that part of the input space and how much of the output it cannot model. This means that the system knows when it is seeing something new that the learning system does not understand. This idea will be carried further in the second part of this thesis, where we explicitly model obstacles as anything the system hasn't seen before.

The imbalance in training data quantities between traversable areas and obstacle areas could lead to concerns that the system will "forget" how to interpret obstacles. This is a common problem with globally optimized functions such as neural networks. For example, if we train a neural network to detect people and then train it on miles and miles of traversable vegetation, it may lose its ability to detect people in its effort to describe vegetation. We avoid this problem by using a local learning algorithm that only makes changes in a local area in the input space. For the above example, the area of the input space that is active from a person would not be changed while it is driving through vegetation. Of course, this requires the various classes that we care about to be separable in the input space, as described in section 3.3.2.

3.4.2 Choosing Features

Feature choice is a difficult problem, and different researchers have proposed various features. Section 4.2 describes the features used in our application, which are a combination of existing features that others have used along with some new features that take advantage of the hit and pass-through density measures that are maintained by our system (see section 1.2). Regardless of the choice of features, the model learning method should handle irrelevant features and determine which features are the most important.

As described in section 3.3.2, this approach requires different types of terrain to be separated in the input space so that the learning system can produce different predictions for each. This assumption is often not met by real data, which is generally noisy and sparse. The second part of this thesis in chapter 6 will describe how introducing spatial correlations can help deal with ambiguous and missing feature data.

Chapter 4

Learning a Terrain Model with Independent Cells

This chapter describes an implementation of the general learning approach presented in chapter 3 to the problem of building a terrain map in vegetation. This approach automatically learns the mapping from features of the sensor data to ground height estimates. Although the system does not directly output obstacle locations, it produces a "supporting surface" estimate at the top of obstacles, so those areas are marked as clearance hazards by the vehicle safety checks performed by the model predictive control system described in section 1.2.

Figure 4.1 shows the steps from sensor data to ground height predictions (this is a subset of the full system diagram given by Figure 3.2 in chapter 3). Each of these steps are described in detail in the following sections. Section 4.1 shows how the system acquires labeled training data for learning, section 4.2 describes the features used, and section 4.3 gives the details of the function approximator we use for learning. Results are given in section 4.4 and a summary and limitations of the approach are given in section 4.5.

Figure 4.1: Steps of learning approach

4.1 Training

Chapter 3 described an online learning approach that allows an autonomous vehicle to collect labeled training data for traversable areas such as that shown in Figure 4.2 simply by driving over representative terrain. As shown in Figure 4.3, the system extracts features for a patch of earth ahead of the vehicle (Cell m) for the purpose of making predictions, but it also stores this feature vector along with its distance from the vehicle when the features

Figure 4.2: Traversable terrain

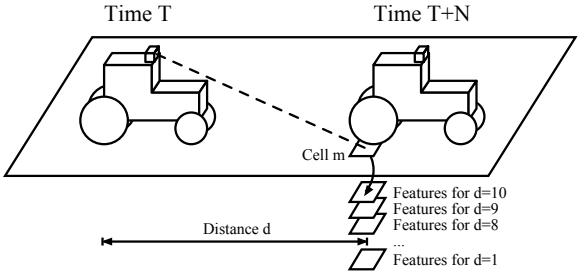


Figure 4.3: Approach for training traversable terrain

Figure 4.4: Untraversable obstacle

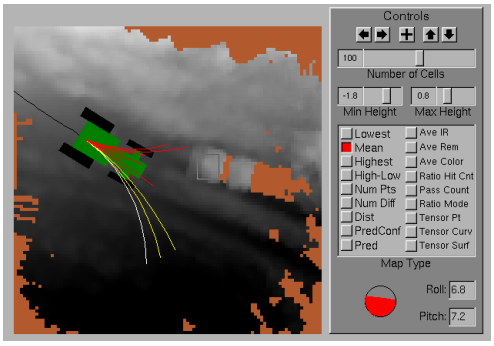


Figure 4.5: Interface showing manual training of untraversable obstacle

were computed. As the vehicle drives towards this area, it recomputes these features using any new data that has arrived, and these feature vectors are also stored. Finally, when the vehicle drives over that patch of earth, it determines the true ground height from the location of its rear wheels, and correlates this true height with the set of feature vectors collected from different distances. This set of feature vectors and its corresponding true height are then given to the learning algorithm to improve its predictions in the future.

The learning approach described in Figure 4.3 allows the system to easily collect massive amounts of training data in representative terrain, but untraversable obstacles like the small vehicle in Figure 4.4 must be trained manually. Figure 4.5 shows our user interface that allows the operator to manually select untraversable obstacles and assign them a true height. The process of storing features as in Figure 4.3 happens automatically for all cells, but only feature sets that have a corresponding true height are used as training data for the learner. This true height can be found automatically by driving over an area, or manually by an operator giving the true height for a group of cells. The true height is often the "highest point" feature for obstacles, but other features such as "lowest point" can also be selected as the manually entered true height for the feature sets from a group of cells.

4.2 Feature Extraction

As described in section 1.2, sensor information is accumulated over time in a global voxel representation. Each voxel is a 15cm cube, and maintains laser hits, laser pass-throughs, and other appearance information for that voxel. The small voxel size is needed because laser pass-throughs in vegetation become more unlikely for larger voxel sizes. We make predictions for each 15cm square cell on the ground, corresponding to a column of voxels. However, the data in a single 15cm wide column of voxels is not enough to get meaningful results for many of the features based on the distribution and shape of points, so we combine multiple cells together and extract features from all the points in a 3x3 neighborhood of voxel columns. The resulting 45cm x 45cm patch on the ground is approximately the same size as the rear wheel contact surface we use to measure true ground height.

4.2.1 Distance

Many features are dependent on how far away from the vehicle they are when they are observed and how many laser range points are in the cell. Figure 4.18 shows a plot of a learned surface and the associated 95% prediction intervals for the difference between true ground height and the 'lowest point' feature using a small dataset taken in vegetation. The plot shows that the performance of using the 'lowest point' feature to predict ground height is dependent on the distance and number of points. The surface was learned using the techniques given in section 4.3. The surface shows that the lowest point becomes a worse predictor of the true ground height at greater distances. The prediction intervals show that predictions using lowest point can be more certain at closer distances and when there are more points in the cell.

Because of observations such as these, we include the number of points in a cell and the distance from the tractor of a cell as features for the learner to use. This allows the learner to automatically find relationships such as those in figure 4.18 to produce better predictions and more accurate prediction bounds.

4.2.2 Simple Statistics on Height

We extract several simple features based on the height of points in a column of voxels, including the mean height and robust lowest and highest point (5% and 95% height value). Figure 4.6 shows the view from the vehicle as it approaches a transition from low grass to tall weeds on relatively flat ground. Figure 4.7 gives the robust highest point feature. The highest laser measurements in the low grass under and behind the vehicle are approximately at ground height and are displayed as a dark gray. The highest points in the tall grass in front of the vehicle and near its right wheels are at the height of the 1m tall grass and are displayed as a light gray. Figure 4.8 shows the robust lowest point feature using the same color scale for height. Comparing Figure 4.8 with Figure 4.7 shows that the laser was able to penetrate the small patches of weeds near the right wheels of the vehicle as well as a short distance into the tall weeds in front of the vehicle. The highest point and lowest point in these areas are very different, and the lowest point gives the true ground height. However, for most of the tall weeded area in front of the vehicle, the lowest point and the highest point nearly match because the laser only gets measurements of the top of

the vegetation. The true ground surface is hidden below these measurements. Since the highest and lowest point features for areas a few meters away from the vehicle match closely for both low grass and tall weeds, but the difference between these features and the actual ground height is very different in these two cases, other features are necessary to allow the learner to discriminate between these two cases and correctly find the true ground height.

4.2.3 Shape of the Point Cloud

Analysis of the eigenvalues of the covariance matrix of the points in a cell gives information about the shape of the point cloud, which can be used to differentiate different terrain types. We incorporate the three shape features discussed in [Vandapel et al., 2004] that are able to separate different types of point clouds (line, plane, scatter). These features are meant to be applied over a neighborhood, so for each set of voxel columns, we find the 3x3x3 cube of voxels with the highest point density and find the shape features from this data. Full classification and grouping of all 3D data as in [Vandapel et al., 2004] could increase performance further.

Figure 4.9 shows the first shape matrix feature, which is also the variance from a plane fit. The laser points in the tall weeds are more scattered than the laser points in the low grass region, so the variance to a plane fit is higher. However, many areas in the tall weeds have low values since the points from the top of the weeds can align in a plane in a similar way to the low grass, especially at greater distances where the laser ray angle is very shallow.

4.2.4 Voxel Hit and Pass-Through Information

Several features are computed from the voxel hit and pass-through information and are useful to determine if an area is solid. In general, areas with a mixture of hits and pass-throughs are vegetation, whereas solid objects have a higher percentage of hits. As a vegetation indicator, we compute the ratio of pass-throughs in voxels with hits to the total number of pass-throughs. To detect solid objects, we compute the percentage of hits for each voxel and then sum the result. Finally, we compute the difference in the number of hits in adjacent columns of voxels to detect vertical solid surfaces (solid obstacles generally have a "shadow" behind them where there are no range points).

Figure 4.10 shows the ratio of pass-throughs in voxels with hits to the total number of pass-throughs. This feature is high when there is tall vegetation that the laser can penetrate, such as the front of the tall weeds section, because there are many pass-throughs in mixed voxels that are likely to be vegetation. The remainder of the tall weeds section shows a mix of high values when the tops of the weeds have varied heights, and low values when the tops of the weeds are fairly constant height.

4.2.5 Appearance

The maximum laser remission (reflectance) value in a cell is used as a feature that can help differentiate between different materials. Figure 4.11 shows that most sections of the tall weeds region have a high remission value.

We can also project values from the infrared camera and color cameras into the grid representation, but these appearance features were not available when most of the experiments in this chapter were performed.

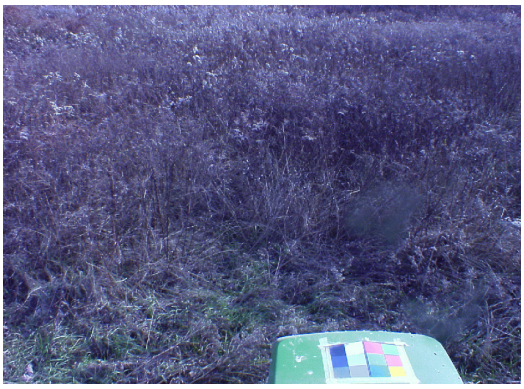


Figure 4.6: View from tractor of grass to tall weeds transition

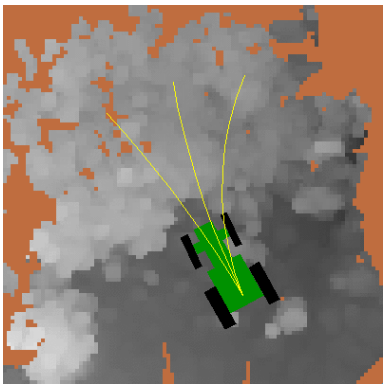


Figure 4.7: Highest point feature

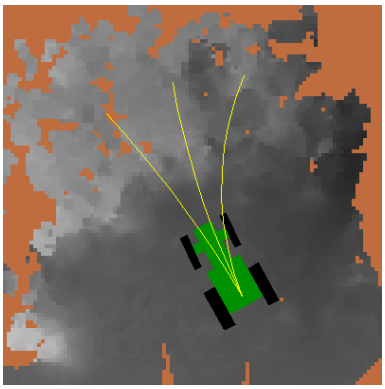


Figure 4.8: Lowest point feature

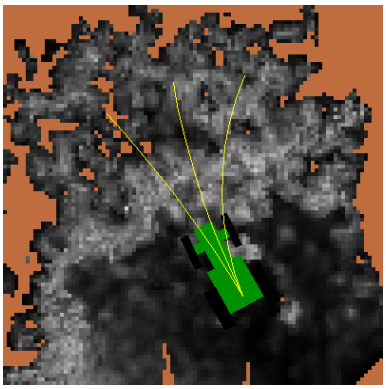


Figure 4.9: First shape matrix feature (variance from a plane fit)

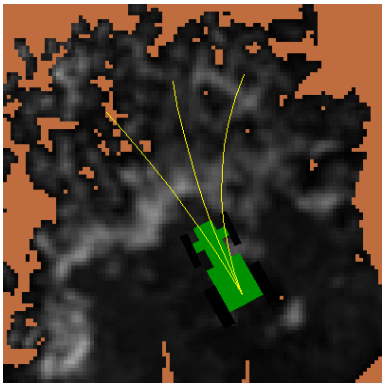


Figure 4.10: Pass-through ratio feature

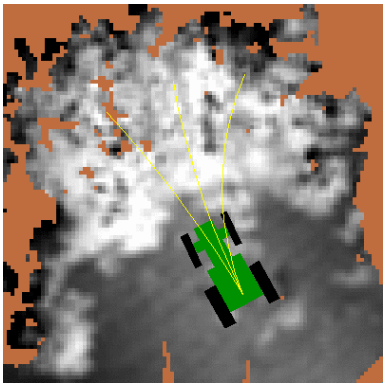


Figure 4.11: Laser remission feature

4.3 Function Approximation (Learning)

Many general function approximators are available, such as neural nets, radial basis functions, locally weighted regression, and many others. A learning method should be able to run online, handle many dimensions (ignoring meaningless dimensions), and produce prediction intervals[1] on its output. We focus on locally weighted regression (LWR) because it provides these capabilities. The rest of this section describes linear regression, locally weighted regression, and an online version of locally weighted regression.

4.3.1 Linear Regression

Linear regression is a common and powerful statistical method for estimating the parameters of a linear model from data. The least-squares estimate of these parameters has its roots in work by Gauss nearly 200 years ago [Gauss, 1809] and details can be found in any introductory statistics textbook.

For a set of input vectors $\mathbf{x}_i$ and output values y_i, a linear model with parameters β can be written as

$$\mathbf{x}_i^T \beta = y_i \tag{4.1}$$

where the constant 1 has been included in $\mathbf{x}_i$ to handle the offset term. Linear regression finds the parameters β that minimize a cost function, which is normally the sum-squared error, since the resulting estimate of β has a number of desirable properties [Ljung, 1987]. Under the assumption that the data came from a noise corrupted deterministic sampling process

$$y_i = f(\mathbf{x}_i) + \epsilon_i \tag{4.2}$$

where ϵ_i are independent with zero mean and variance σ^2, and the assumed linear structure of the function f is correct, then the estimate of β is unbiased and will converge to the correct parameters with probability 1 as $N \to \infty$. If the disturbances ϵ_i have a Gaussian distribution (σ^2 unknown), then the estimate of β has a multidimensional t-distribution, which can be projected into the output space to find prediction intervals for the prediction of a new query $\mathbf{x}_q$.

Figure 4.12 shows an example of linear regression. The least-squares estimate of the line from the noisy data closely matches the true linear function. 95% prediction intervals are also shown. Predictions are most certain at the center of the graph, where there is the most data support. Given that the assumptions of the model are satisfied in this case, it is expected that the estimate does well.

Figure 4.13 shows a nonlinear sigmoid function and the resulting least-squares linear estimate. The estimate succeeds in finding the general trend of the data, and the prediction intervals correctly represent its lack of predictive capability, but a global linear model does not have the representational power to capture the nonlinear nature of the true function.

[1] Prediction intervals give the range where new data points are expected to be and include both the uncertainty in the data and the uncertainty in the learned model. These are more useful in our domain than confidence intervals, which give the range where the model parameters are expected to be and only include the uncertainty in the learned model, since we are ultimately more interested in the quality of the learned model's predictions than in the quality of its parameters.

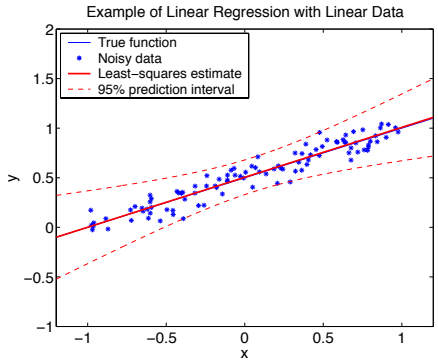


Figure 4.12: Linear regression for a linear model

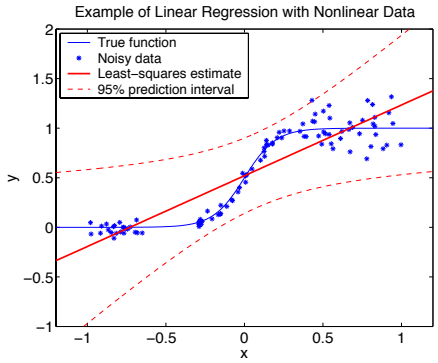


Figure 4.13: Linear regression for a nonlinear model

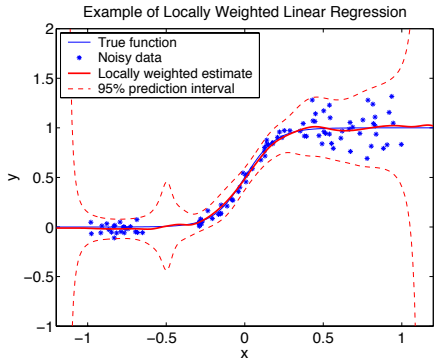


Figure 4.14: Locally weighted linear regression for a nonlinear model

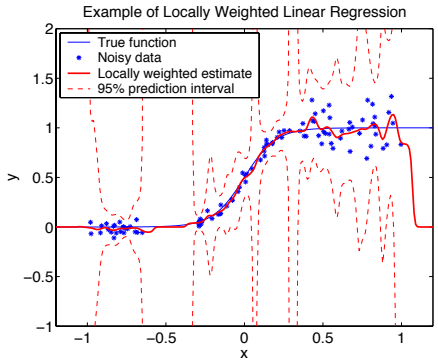


Figure 4.15: Locally weighted linear regression for a nonlinear model with incorrect bandwidth

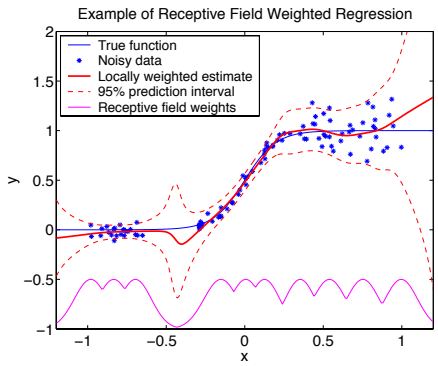


Figure 4.16: Receptive field weighted regression for a nonlinear model

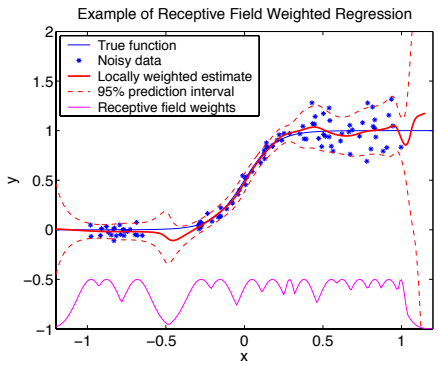


Figure 4.17: Receptive field weighted regression for a nonlinear model with incorrect initial receptive field size after training on the data for 20 iterations

4.3.2 Locally Weighted Regression

To be able to represent nonlinear functions while maintaining the computational and theoretical advantages of linear regression, a number of researchers have investigated locally weighted regression (LWR) [Atkeson et al., 1997] [Schneider and Moore, 2000]. LWR uses a kernel function such as a Gaussian to weight the data points differently, allowing nearby data to have more influence on a prediction than data that is far away. Training is trivial in LWR, and simply involves the addition of a new data pair $(\mathbf{x}_i, y_i)$ into memory. Then, when a prediction is desired for a new query point $\mathbf{x}_q$, a *local* linear model centered at the query point is constructed just as in normal linear regression, except that the data points in memory are weighted by their distance to the query point through the kernel function. This process can model arbitrary functions, but uses the bandwidth of the kernel function as a bias towards smooth functions that look linear (at least locally). The choice of bandwidth has a large effect on LWR, and it is usually chosen using leave-one-out cross validation, which can be performed very efficiently. [Schaal and Atkeson, 1994] has converted the prediction intervals from linear regression into a locally weighted version that only uses the data that is near the query point to determine confidence. The size of the prediction bound depends both on the density of data points in the area, and on the noise in the outputs of nearby data points that cannot be explained by the model. These bounds are only correct if the LWR estimate is unbiased at the query point, which is usually difficult to determine, but if the bandwidth is correct then the true function should be linear in the region around the query point so this is a weak assumption.

Figure 4.14 shows the result of using LWR to find the nonlinear sigmoid curve from figure 4.13. A global bandwidth parameter was found using leave-one-out cross validation. The curve matches the true function well, and the prediction intervals show useful information. The intervals show that the estimate is unreliable to the left and right where there is no data, as well as being less reliable around $x = -0.4$ where there is a gap in the data. They also show reduced confidence on predictions for the area on the right where the data points have more noise.

Figure 4.15 shows the result of a poorly chosen kernel bandwidth, where the function fits the noise. In this case, only very nearby data points are used for the regression, so the result is similar to nearest-neighbor, which performs poorly with noise. The prediction intervals are very wide because each local regression is performed with just a few points so it has little confidence in its result. If the bandwidth is chosen poorly in the other direction, all the data points contribute to the regression, and the result will approach global linear regression as shown in figure 4.13.

There is another problem with memory based locally weighted regression. All the data is stored in memory, so as more data is collected, the predictions become slower. LWR has been implemented using K-d trees [Atkeson et al., 1997], which can find nearby points very quickly. However, this does not solve the problem that predictions become slower as more data is collected. In our application, the learning system continually collects data to improve its models so storing all the data in memory is not feasible.

4.3.3 Receptive Field Weighted Regression and Locally Weighted Projection Regression

[Schaal and Atkeson, 1998] has extended LWR to work online with a continuous stream of data using a technique called receptive field weighted regression (RFWR). The essential point of RFWR is to build up a number of locally linear models during training that maintain sufficient statistics about the actual data points. Once this is done, the data can be thrown away and new predictions are made from a weighted combination of linear models instead of the data points directly. A forgetting factor is used to slowly discount old experience as it is replaced with new data. Each local linear model maintains a region of validity called a receptive field.

Figure 4.16 shows the result of using RFWR to fit the sigmoid. The bumps at the bottom of the plot show the strength of the receptive fields for each individual local model. RFWR maintains a covariance matrix on the regression parameters for each local model as well as estimates of the noise of the data, so prediction intervals can be computed as shown in the figure.

RFWR uses gradient descent to locally optimize the size of the receptive fields. Local optimization is important for online implementation because adding a single data point only affects a small number of receptive fields. RFWR adds new fields when there is no current receptive field at the location of a new data point, and it prunes fields if they overlap too much with their neighbors. Figure 4.17 shows the local optimization of the receptive fields. The same incorrect kernel bandwidth used in figure 4.15 was used to initialize new fields. After being trained for a single iteration on the data, there were 35 receptive fields and the result was overfitting similar to figure 4.15. However, by continuing to present random permutations of the same data, the algorithm pruned many fields and made the remaining ones larger to produce the result shown in figure 4.17 with 15 receptive fields after 20 iterations.

More recently, this algorithm has been extended to also handle high dimensional spaces (including redundant and irrelevant dimensions) with locally weighted projection regression (LWPR) [Vijayakumar and Schaal, 2000] [Vijayakumar et al., 2005]. This technique uses a hierarchy of projections in the input space to fit a local model to a low-dimensional manifold embedded in the high-dimensional space. For the single-dimension example figures in this section, this algorithm gives results similar to RFWR shown in figures 4.16 and 4.17. As with LWR, the size of the prediction bound depends both on the density of data points in the area, and on the noise in the outputs of nearby data points that cannot be explained by the model [Vijayakumar et al., 2005]. Our system uses LWPR for its online learning mechanism.

4.3.4 Comparison to Other Learning Algorithms

Other general function approximators can also fit nonlinear functions, but LWR and its variations have several important advantages that are due to their local nature. Because LWR maintains a database of training examples (or a set of local linear models), it does not suffer the same interference problems that globally optimized techniques such as neural networks experience when previously learned relationships are “forgotten” as the net tries to accommodate new data. Neural networks also have difficulty with local minima during

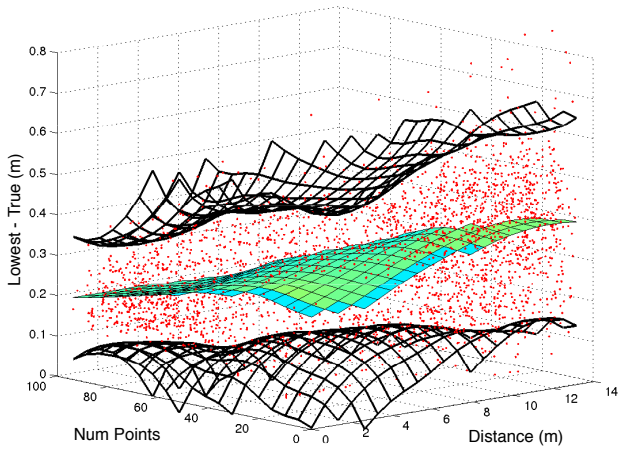


Figure 4.18: Learned surface and prediction intervals showing that the performance of using the lowest point to predict ground height is dependent on distance and number of points.

the complex global optimization process. LWR evades these problems by only looking at local problems that are easy to optimize locally. The statistical foundations of LWR also allow straightforward computation of prediction intervals.

Traditional problems with locally weighted learning include prediction times that are dependent on the number of data points in memory, and a difficultly with high dimensional spaces. The LWPR algorithm [Vijayakumar et al., 2005] described in the previous section has addressed both of these problems by maintaining a set of local models instead of the actual data points, and by performing local dimensionality reduction. The dimensionality reduction of LWPR assumes that the input data actually lies on a low-dimensional manifold embedded in a higher dimensional space. Therefore the algorithm becomes slower as the input data fills more of the input space. In our experiments, the algorithm slows down during initial training as it adds new receptive fields to cover the input data, but then it stabilizes once the relevant parts of the input space are covered. The system produces estimates for the area in front of the vehicle at approximately 1Hz, but most of this computation time is spent calculating features (especially the point cloud shape features which require singular value decomposition).

4.3.5 LWPR Example

Our online system uses LWPR as its function approximator. Figure 4.18 shows an example of LWPR for a simple two dimensional problem. The inputs are the number of points in a column and the distance of the column from the vehicle, and the output is the adjustment to the lowest point in the column to get the true ground height. The plot shows the training data points, the learned surface, and the prediction intervals. This example shows that at closer distances and higher numbers of points, the lowest point in a column is a better predictor of ground height (the adjustment is smaller), and the confidence in that prediction is higher (the prediction intervals are narrower).

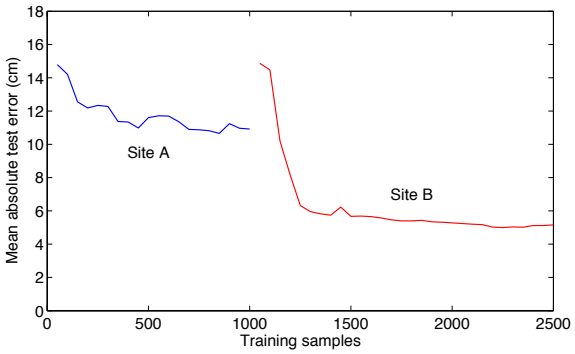


Figure 4.19: Adapting to Environmental Change

4.4 Results

We have tested the system described in this chapter at a nearby working farm and an undeveloped area with tall weeds. Results show that the system can find obstacles in sparse vegetation and improve predictions of vehicle safety quantities such as roll. We also show the benefits of adapting online when the vehicle encounters challenging new terrain and give an example that includes prediction intervals on its output. These results use only the laser features described in section 4.2, except for the prediction intervals result in section 4.4.5, which also uses color features.

4.4.1 Adapting to Environmental Change

To test the ability of the system to automatically adapt to a change in the environment, we collected data from two test sites and split the data from each site into a training and test set. Site A consisted of challenging dense vegetation approximately 0.75m high, and Site B had sparse vegetation of heights up to 1m high as well as gravel roads without vegetation. The training data from Site A was presented to the learning algorithm, and the prediction error on the corresponding test set from Site A was periodically computed. Figure 4.19 shows the algorithm learning the characteristics of this site and reducing prediction error on the test set. Then, starting with sample 1000, the system was presented with data from Site B. After being trained in the first site with dense vegetation, the algorithm initially did poorly in the new site with roads and more varied vegetation. However, the learner quickly adapted to the new environment and ended up with a prediction error similar to an experiment that only trained at Site B.

Although not shown in Figure 4.19, if the system returns to Site A after sample 2500, the prediction error becomes slightly higher than the error at sample 1000 at the end of the first training session in Site A. This means that despite the local nature of the learning algorithm, there is still some interference between these two training sites. A richer feature set that is better able to separate these two sites in feature space may perform better. However, even if the training from a previous environment does not help in a new environment, the system is still able to adapt to the current conditions and perform well in the new environment.

Figure 4.20: View from tractor of person kneeling next to sparse vegetation

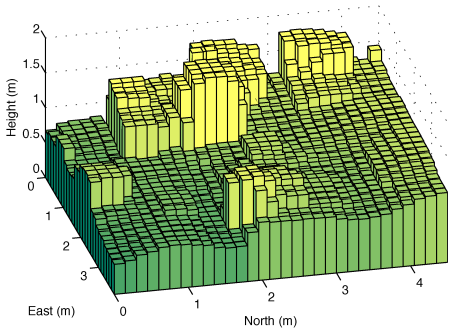


Figure 4.21: Highest point surface

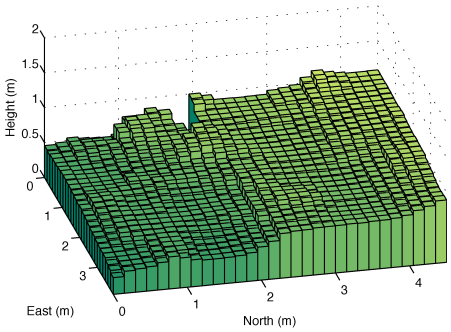


Figure 4.22: Lowest point surface

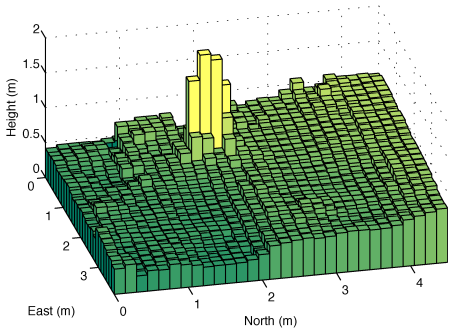


Figure 4.23: Learned result removes the vegetation but keeps the person

4.4.2 Obstacle in Vegetation

Figure 4.20 shows an example of a person kneeling among tall sparse vegetation. The angle of Figure 4.20 makes it difficult to see the taller weeds, but the highest point statistic in Figure 4.21 shows that the kneeling person is beside various vegetation of a similar height. Because the vegetation is sparse, the laser was able to penetrate through the tall weeds and the lowest point feature in Figure 4.22 gives a good representation of the ground plane, but it makes the person disappear along with the vegetation. From these figures, it is clear that using a simple statistic such as the highest or lowest point will lead to false positives or false negatives in the presence of vegetation.

Instead of hand-crafting a set of rules and thresholds for how to combine our features to accomplish this task, we used the learning method described above to find the mapping automatically. We drove around in similar types of vegetation for approximately 10 minutes to let the system learn what types of features represent compressible objects. To teach it

Figure 4.24: Person kneeling behind similar height sparse vegetation

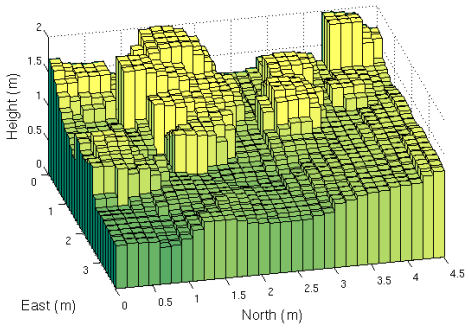


Figure 4.25: Highest point surface

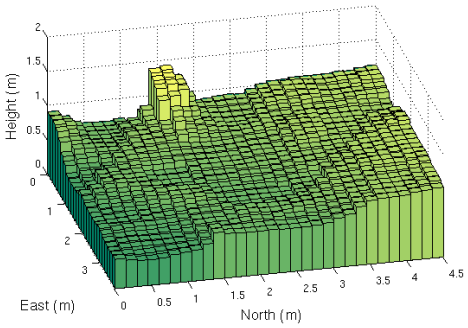


Figure 4.26: Lowest point surface

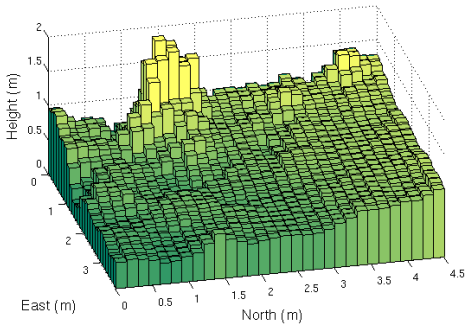


Figure 4.27: Learned result removes the vegetation but keeps the person

about solid objects, we drove up to a set of walls, posts, and hay bales and then manually selected them to be trained using the highest point as the truth value.

After this simple training procedure, the learning system was able to produce the results shown in Figure 4.23. The vegetation has been removed, thus reducing false positives, but the person remains and would trigger a clearance hazard.

Figure 4.24 gives a similar example, except this time the person is hidden behind the vegetation. The person is sitting back on his feet, so the lowest point feature in Figure 4.26 recovers the person's lap, but this may not be tall enough to trigger a clearance hazard. The system output in Figure 4.27 again shows that the system has learned how to discriminate between sparse vegetation and a solid obstacle. This example is more challenging than the previous one in Figure 4.20, and the dense bases of the vegetation in front of the person and at the left edge are not removed. However, a vehicle using this result to navigate would still choose to drive through the vegetation and avoid the person.

Figure 4.28: Tractor driving through weeds on a slope

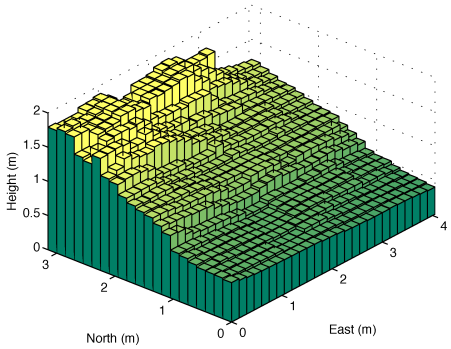


Figure 4.29: Lowest point surface

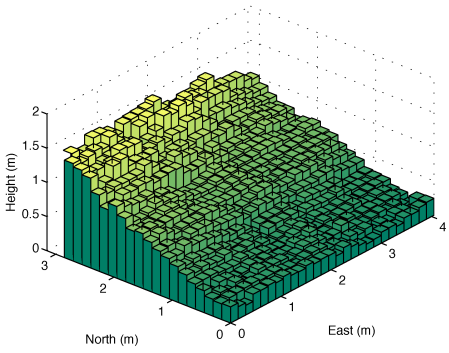


Figure 4.30: Learned result removes the vegetation and retains true slope

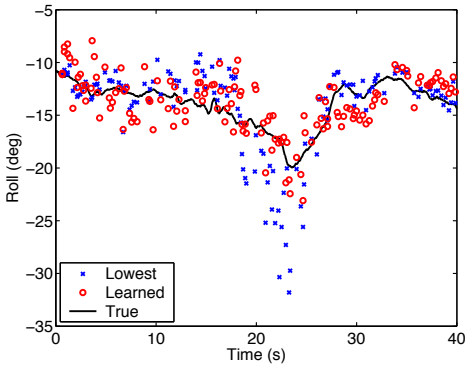


Figure 4.31: Learned result produces more accurate roll predictions than using lowest point

4.4.3 Roll Predictions in Vegetation

This experiment further investigates the effects of ground estimate quality on vehicle navigation. As described in section 1.2, the local navigation system uses ground height estimates to predict if the vehicle roll and pitch will be within safety constraints for a given control.

Figure 4.28 shows the vehicle traversing a slope with vegetation. Using the lowest point as shown in Figure 4.29 results in overestimating the ground surface in some of the taller vegetation on the left. After training our system by driving it through some similar vegetation for approximately 10 minutes, it produces more accurate height predictions as shown in Figure 4.30. From the system level, we really care about predictions of safety parameters such as roll and pitch. Figure 4.31 shows that the improved height estimates result in better roll predictions when compared to the true value found when we drove over that area. The terrain sections shown in Figures 4.29 and 4.30 were used for the roll

predictions 20 seconds into the test and explain the poor performance using the lowest point.

If the vegetation was on the bottom of the slope instead of the top, then making this mistake could be dangerous because the system would believe that the vehicle could drive on top of the vegetation instead of predicting that it would drop below the vegetation and perhaps cause a roll-over hazard. Correct ground height estimates result in correct navigation decisions.

4.4.4 Online Adaptation

We also performed a set of experiments in dense vegetation that the laser cannot penetrate beyond a short distance to understand the benefits of adapting online. To make predictions ahead of the vehicle in these circumstances, the system must use its prior experience with similar terrain to determine an estimate of the ground plane. Figure 4.32 shows the tractor entering dense vegetation over 1m tall. The three graphs in Figures 4.33 to 4.35 show the results of three different ways to approach this difficult task. Each graph shows the error from the true ground height of the lowest point feature and the predicted ground height at a distance approximately 5m in front of the vehicle as it drives into the tall vegetation. At the beginning, when it is driving through relatively short grass, the lowest point feature works well, but after 10 seconds into the run when the vehicle enters the tall vegetation, the laser cannot penetrate the vegetation 5m in front of the vehicle, and the predictions must adjust.

The first case in Figure 4.33 shows a system that was trained in other vegetation in a nearby area for approximately 10 minutes, but then the learning algorithm is turned off during the test. Using only its prior experience it is able to handle the transition because it has learned to discriminate between the low grass and the tall vegetation. However, the vegetation that it was trained in was a different height than this vegetation, so there is a fairly constant offset between the predicted ground height and the actual ground height, which would result in many false positives for the system.

Figure 4.34 shows the predictions of the system when it is started fresh with no prior training but is allowed to adapt during the run. Without any experience, it fails to recognize the transition to weeds and continues using the lowest point. However, after it enters the weeds and collects some data, it adapts its predictions to match the vegetation height. Due to its limited training, it continues to make errors about when to use the lowest point and when to drop below it, but the adaptive capability of the system will allow it to get better over time. The areas in Figure 4.34 where the system adjusts the lowest point match very well with the true value. However, since it has just started training, it has essentially overfit all features to that offset, and when it comes out of the tall weeds and re-enters the short grass, it incorrectly lowers that area by this offset as well. Only after training for some time in both types of vegetation can it correctly switch between the two.

The third case in Figure 4.35 shows what happens when the vehicle was trained just as in the first case in Figure 4.33, but then is allowed to continue adapting as it drives into this new vegetation. Its prior training allows it to handle the transition from short grass to tall weeds, and then it adapts to the new vegetation height so that it doesn't have the constant offset like in Figure 4.33.

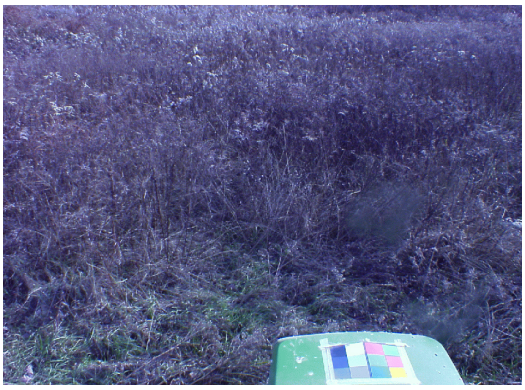


Figure 4.32: View from tractor of grass to tall weeds transition

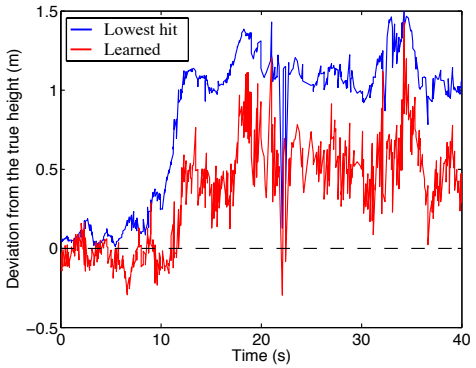


Figure 4.33: Learned result without adaptation

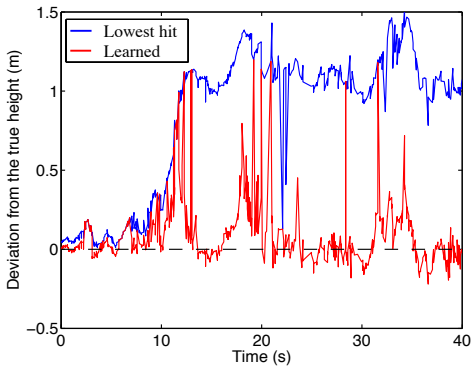


Figure 4.34: Adaptation without previous learning

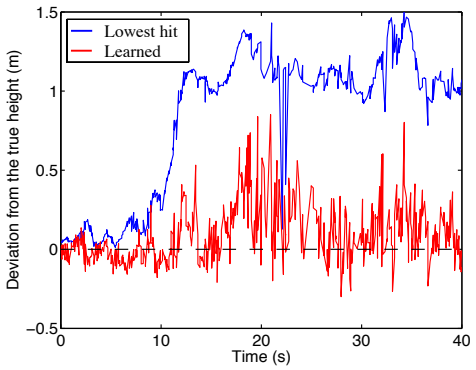


Figure 4.35: Learned result with online adaptation

These results show the benefit of adapting online to the current conditions, but they also show a weakness of the approach. The system makes predictions by looking at a single grid cell in isolation. Without the context that humans use so effectively, the system is at a great disadvantage because a patch of short grass and a patch of tall grass may have very similar laser feature signatures, especially at a distance. Continually adapting online is one way to give the system context by exploiting the local similarity of the world. However, even with online adaptation, the ground estimates in tall vegetation shown in Figure 4.35 are very noisy and would cause false positives during autonomous navigation.

One of the reasons that the ground height estimates are so noisy is that the system makes the strong assumption of independence between neighboring cells. Real ground heights are highly correlated spatially, but the system does not take this into account, so the ground height estimates do not resemble actual ground. The estimates could be smoothed, but a blind smoothing operation may also smooth out a true obstacle like the kneeling person in

section 4.4.2. Chapter 6 describes a system that relaxes this independence assumption and produces smooth ground estimates while still finding positive obstacles.

4.4.5 Prediction Intervals

As described in section 4.3, the learning algorithm produces prediction intervals on its output. This section shows the system output for a transition into tall vegetation similar to the previous section, but includes prediction intervals on the result. The system used for this test includes color and infrared data as features in addition to the range based features used in the previous experiments.

Figure 4.36 shows the view from the tractor of a transition from low grass to tall dense weeds. The tall weeds have a strong yellow color, so including color in addition to range based features better separates the different terrain types in the input feature space.

After the system was trained in similar terrain for a little over 10 minutes, it produced the ground height predictions shown in Figure 4.37. The same predictions are shown from the side with the sensor data in Figure 4.38. Figures 4.40 and 4.41 show cross-sections of these predictions compared with the lowest hit or pass feature and the true ground height found when the vehicle drove into the tall weeds. Note that Figures 4.40 and 4.41 show a snapshot of the predictions at different distances in front of the vehicle (measured from the rear wheels) at a given time, unlike the figures in section 4.4.4 which show predictions for a constant distance in front of the vehicle over time as the vehicle drove.

These results show that the system gives fairly accurate ground height estimates in the low grass near the vehicle, but does not apply enough of an offset to the lowest point in the tall dense weeds, which would likely cause a false positive for the system as it tried to enter the weeds. The system applied an offset based on its expected vegetation height for those features based on its training data. As in the previous dense vegetation example in section 4.4.4, the system's expected vegetation height from its training data does not match this test case, so the ground predictions are wrong.

The system produces prediction bounds on its estimates, which could be used to slow the vehicle down in areas of high uncertainty, or avoid such areas completely. Figure 4.39 shows the ground surface and $\pm 3\sigma$ prediction bounds for this case. The prediction bounds are also displayed in Figures 4.40 and 4.41. The true ground height is usually within these bounds.

Figure 4.40 shows high uncertainty 5m in front of the rear wheels even though the predictions are correct. This is the front part of the vegetation that the laser can penetrate. Further in front of the vehicle, the laser does not penetrate the vegetation and the ground estimates have a constant offset from true ground height, as discussed above. In both Figures 4.40 and 4.41, the system becomes less confident in its predictions for farther distances where it has less data.

Figure 4.36: View from tractor

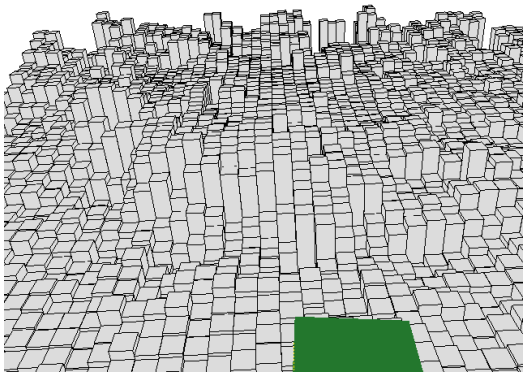


Figure 4.37: Learned height predictions

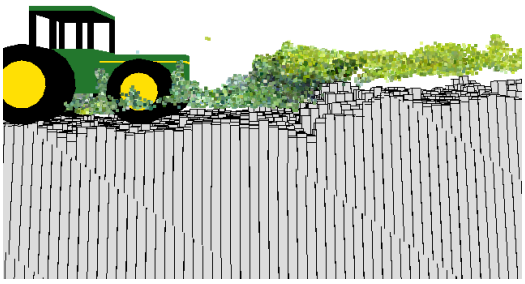


Figure 4.38: Side view showing ground predictions and input data

Figure 4.39: Side view showing $\pm 3\sigma$ prediction intervals

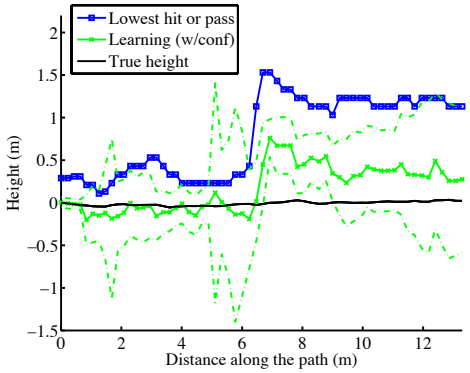


Figure 4.40: Height prediction comparison for left wheel

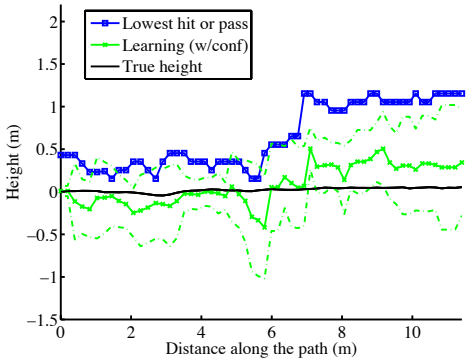


Figure 4.41: Height prediction comparison for right wheel

4.5 Summary and Limitations

This chapter has described a system that produces improved predictions of vehicle safety parameters by learning predictions of the load-bearing surface in vegetation while still finding positive obstacles. The entire system runs online on our autonomous tractor test platform, and can improve and adapt over time. A key benefit of this approach is that the vehicle can generate labeled training examples of features at different distances just by driving through an area of interest. This makes it easy to collect massive amounts of training data quickly, which is important since one of the main difficulties in applying machine learning techniques to outdoor vehicle navigation is the lack of labeled training data [Dima et al., 2004a].

However, the results in dense vegetation still produce many false positives, the system explicitly learns a vegetation height so it is not general across vegetation of the same type with varying heights, and non-traversable obstacles must still be manually labeled. These limitations served as motivation for the spatial model approach presented in chapter 6, and are discussed below.

4.5.1 Independence Assumption

The strong assumption of independence between cells is limiting because it prevents the system from including spatial context that could disambiguate cells in different types of terrain that have similar features, as shown in Figure 4.42. Range features are often similar between low grass and the top of tall dense vegetation, especially at farther distances. This results in uncertain predictions from the learner in these cases. The noisy predictions in Figure 4.35 and the large prediction intervals in Figure 4.40 are evidence of this occurring.

One way to reduce this problem is to expand the feature set to better separate the input features of different terrain types. However, this makes the learning problem more difficult by increasing the dimensionality, and it also requires good features. Quite a bit of feature engineering went into this approach to be able to separate different object types.

Chapter 6 presents a method for including spatial correlations in the terrain model to disambiguate areas with similar features. This method includes a 3D model structure that uses the data directly instead of extracting complicated features. The spatial correlations help smooth the ground estimates and also help fill in predictions in areas with little or no data.

4.5.2 Explicit Vegetation Height

This approach learns the mapping from features extracted from a column of data to the supporting ground surface height. For dense non-penetrable vegetation, this results in the system learning a typical vegetation height as a function of the input features which is then used as an offset below the lowest hit feature for ground predictions. Because the system is trained explicitly on the height of vegetation, it can perform poorly when it first encounters vegetation of a new height or if there is vegetation of different heights with similar features. These problems were shown in the experiments in sections 4.4.4 and 4.4.5.

Chapter 6 presents an alternative method that learns the properties of voxels instead of columns and then tries to infer the location of voxel type transitions such as ground to

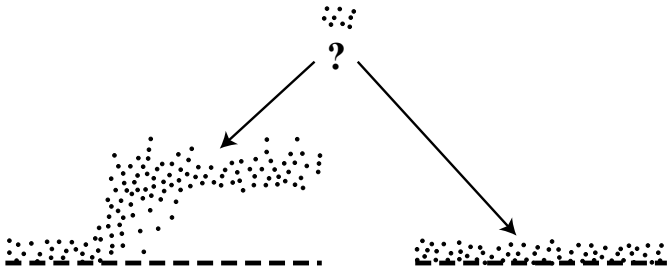


Figure 4.42: Data from different types of terrain such as tall weeds on the left or low grass on the right may appear similar within a single terrain patch

vegetation or vegetation to free-space. In this way, the system is able to infer vegetation height from the data in the local area instead of training on it explicitly.

4.5.3 Obstacles

This approach requires non-traversable obstacle features to be trained manually. For the experiments that used laser features to discriminate solid obstacles from sparse vegetation, this process added some time to the training process but was feasible because the features of the expected obstacles were similar. However, when appearance features such as color are added to the system, it is not clear how to train the system to handle obstacles because of the wide range of possible feature values. Active learning methods could help select training examples [Dima et al., 2004a], but there still is the problem of finding example obstacles that fill the space of input features for the many obstacles that could possibly be encountered.

As described in section 4.3, the learning algorithm used in this approach produces prediction intervals on its output that are based on the output variation it can't explain and the amount of data support for that area of the input space. This means that the prediction intervals are very large for unknown areas, which could be a good indicator for obstacles.

The approach presented in chapter 6 exploits this idea by training on expected substances such as vegetation or bare ground and treating anything else as an obstacle.

Chapter 5

Markov Models

The assumption of temporal or spatial independence is often overly restrictive because it ignores important correlations and it prevents the exploitation of prior knowledge of dependencies in how these quantities change in time or across space. This chapter describes various models and techniques for including these dependencies, and chapter 6 applies these ideas to the problem of terrain modeling.

A common approach for relaxing the independence assumption and yet still remaining computationally tractable is to assume that the stochastic process of interest obeys the Markov property. This assumption says that if we know the present state, then knowing past states gives no added information about future states. Given the present, the future is conditionally independent of the past.

$$P(X_{t+1} \mid X_t, X_{t-1}, \dots) = P(X_{t+1} \mid X_t) \tag{5.1}$$

This implies that the current state X_t captures all the relevant information from the past and that the path of how the system reached that state is unimportant for predicting the future. Although the state X may need to be prohibitively large for an actual system to obey the Markov property, many models and techniques using this assumption with a simplified state have had great practical success by adequately capturing the most important relationships in the physical quantities being studied.

The Markov property can apply to spatial relationships as well as time dependencies. For a one dimensional Markov chain along the discrete spatial index i,

$$P(X_i \mid \mathbf{X}) = P(X_i \mid X_{i-1}, X_{i+1}) \tag{5.2}$$

where $\mathbf{X}$ is the set of all states. In higher dimensions, we can define a neighborhood N_i of variables such that

$$P(X_i \mid \mathbf{X}) = P(X_i \mid X_{N_i}) \tag{5.3}$$

As in the time domain, these assumptions allow spatial correlations and contextual information to be included in a tractable way.

When using Markov models for modeling a physical phenomenon, we generally only have access to noisy observations Y of the state X, which results in a *hidden Markov model*, as shown in Figure 5.1. In addition to the Markov property between states X, this model

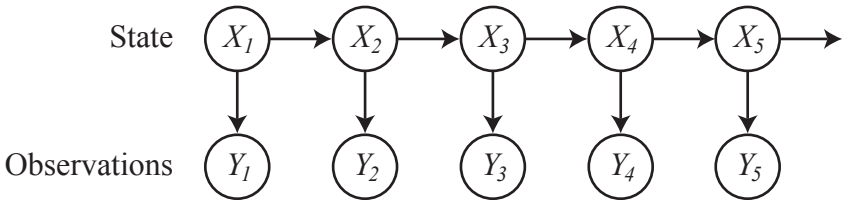


Figure 5.1: Graphical model of a hidden Markov model (or Kalman filter), showing the dependencies between states X_{i-1} and X_i and between states X_i and observations Y_i

makes the further assumption that the observations are conditionally independent given the current state.

$$P(Y_i \mid \mathbf{X}, \mathbf{Y}) = P(Y_i \mid X_i) \tag{5.4}$$

As above, this implies that the state X_i captures all the relevant information of the system, and it further implies that any measurement noise is independent given the state.

A hidden Markov model encapsulates two separate components, a transition (process) model, and an observation (measurement) model. The transition model describes the relationship between neighboring states (in time or space) and is shown graphically by the horizontal links between X_i's in Figure 5.1. The observation model describes how observations are generated from states and is shown graphically by the vertical links between X_i and Y_i in Figure 5.1.

Different algorithms are used with these models depending on the structure of the model, the types of transition and observation models, and the desired variables that are being estimated. Estimating the state X from observations Y using a known transition and observation model is described as inference or state estimation. Using data to find the transition and observation models is generally called learning or parameter estimation.

The following sections describe some common model structures and their associated inference algorithms for state estimation. Although varying substantially in form, each algorithm uses an assumed model (perhaps learned from data) and noisy observations to find estimates of state variables of interest. Chapter 6 describes how these model structures can be used to represent natural assumptions about terrain.

5.1 Markov Chains and Hidden Markov Models

Hidden Markov models (HMM) have been successfully used for a wide variety of temporal and spatial modeling problems, from position estimation [Maybeck, 1979a] to speech recognition [Rabiner, 1989] to protein modeling [Krogh et al., 1994]. The discrete steps in space or time of these models (Figure 5.1) make them especially appropriate for computer modeling and sampled data filtering. Although the details of the state estimation algorithms depend on whether the states are continuous or discrete, the probabilistic inference procedure follows the same steps since the underlying independence assumptions represented by the graphical model in Figure 5.1 are the same.

Figure 5.2 shows two common estimation problems. In each case, we are trying to estimate the state at X_t, but the filtering problem only uses past and current data, whereas

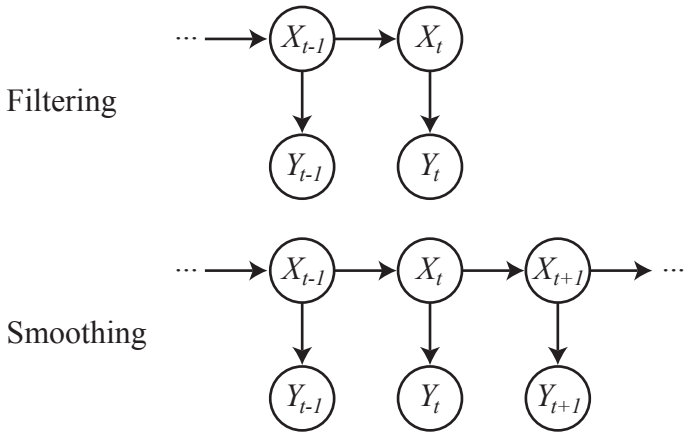


Figure 5.2: *Filtering* uses past and current data to estimate X_t. *Smoothing* uses past, current, and also future data to estimate X_t.

the smoothing problem also uses future data to improve the estimate. The Markov property described above allows us to decouple these problems.

$$\textit{Filtering} \qquad P(Y_{1:t}, X_t) = P(Y_{1:t-1}, X_t)P(Y_t \mid X_t) \tag{5.5a}$$

$$\textit{Smoothing} \qquad P(Y_{1:T}, X_t) = P(Y_{1:t-1}, X_t)P(Y_t \mid X_t)P(Y_{t+1:T} \mid X_t) \tag{5.5b}$$

These equations lead to efficient recursive algorithms for both discrete and continuous state hidden Markov models (this is explored in more detail in [Minka, 1998]). The following sections describe the continuous state Kalman filter and Kalman smoother, and the discrete state hidden Markov model and hidden semi-Markov model.

5.1.1 Kalman Filter

The Kalman filter [Kalman, 1960] [Maybeck, 1979a] is a recursive algorithm for optimally estimating the state of a linear system with Gaussian noise corrupted measurements. It has found great practical value and is the workhorse for position estimation and many other state estimation tasks.

A key feature of the algorithm is its recursive nature, allowing a system to efficiently run online and produce state estimates using current sensor measurements. The graphical model in Figures 5.1 and 5.2 shows the conditional independencies that allow this recursive estimation to take place. For example, at time t in the 'filtering' part of Figure 5.2, we may want to estimate the state X_t. The graphical model shows that if we know X_{t-1} then X_t is conditionally independent of the previous states and observations so they do not need to be stored. Before the measurement Y_t arrives, the transition model can be used to *predict* the value of X_t using the previous state X_{t-1} which encapsulates all the information from the previous measurements. Once the measurement Y_t arrives, the observation model can be used to *update* the estimate of X_t by combining the prediction with the actual sensor data. The *predict* and *update* steps relate to the two components in equation 5.5a. If the transition and observation models are linear and the process and measurement noise are Gaussian, the Kalman filter finds the optimal estimate of X_t by performing these two steps.

For a linear model with Gaussian noise, the state estimate is also Gaussian and can be represented by its first two moments, its mean X and covariance Q (the state covariance of a Kalman filter is generally represented as P in the literature, but Q is used here to prevent confusion with the probability operator $P(\cdot)$ used in this chapter). In the following equations, the state estimate X and its associated covariance estimate Q for a given time step t_i are shown before the measurement update as $X(t_i^-)$ $Q(t_i^-)$ and after the measurement update as $X(t_i^+)$ $Q(t_i^+)$ which represents the optimal estimate and covariance at that time step. Measurements Y also have an associated covariance R. The matrix Φ represents the state transition model and the matrix H is the observation model. The state transition matrix Φ maps states to states at the next time step, whereas the observation matrix H maps states to measurements and is used in the following equations to give the expected measurements for a given state.

Prediction

$$\begin{aligned} X(t_i^-) &= \Phi X(t_{i-1}^+) \\ Q(t_i^-) &= \Phi Q(t_{i-1}^+)\Phi^T \end{aligned} \tag{5.6}$$

Update

$$\begin{aligned} X(t_i^+) &= X(t_i^-) + K\left(Y_i - HX(t_i^-)\right) \\ Q(t_i^+) &= Q(t_i^-) + KHQ(t_i^-) \end{aligned} \tag{5.7}$$

Kalman Gain

$$K = Q(t_i^-)H\left[HQ(t_i^-)H^T + R\right]^{-1} \tag{5.8}$$

Equation 5.8 is known as the *Kalman Gain*, which controls the weighting between the prediction and the measurement update. The Kalman gain is determined by the ratio of the state covariance Q and the measurement covariance R. As shown in equations 5.7 and 5.8, when the predicted state has a high uncertainty (high Q) relative to the measurement uncertainty R, the Kalman gain K is high and the predicted state $X(t_i^-)$ is moved strongly in the direction of the residual error $(Y_i - HX(t_i^-))$ between the expected measurement and the actual measurement to get the optimal estimate. Conversely, if the predicted state uncertainty is low relative to the measurement uncertainty, then the Kalman gain will be low, resulting in the measurement having less of an effect (it is more strongly filtered).

The Kalman filter is only optimal for linear Gaussian models, but many extensions allow it to be applied to nonlinear problems with more complicated distributions. To handle nonlinear process models, the extended Kalman filter [Maybeck, 1979b] linearizes about the current state estimate and then applies the standard Kalman filter equations (see positioning example in appendix A.2), the assumed density filter [Maybeck, 1979b] projects the true density to an assumed form that is easier to handle, and the particle filter [Gordon et al., 1993] uses a set of samples to represent the distribution. Many other extensions exist.

5.1.2 Kalman Smoother

The Kalman filter is well-suited to online estimation applications since it provides the best estimate of X_t given all the measurements that have been received up until time t. However, if measurements beyond time t are available as in the 'Smoothing' part of Figure 5.2, then a better estimate for X_t can be computed by using all the measurements.

The Kalman smoother [Maybeck, 1979b] uses a combination of two Kalman filters, one running forward in time that incorporates $Y_{1:t}$ and one running backward in time that incorporates $Y_{t+1:T}$, to find the optimal estimate of X_t given all the measurements $Y_{1:T}$. This decoupling is evident in equation 5.5b. In the Kalman smoother framework, each of the two Kalman filters produces an estimate of X_t and its associated covariance Q, and then they are optimally combined by weighting each according to their confidences Q. An equivalent interpretation is that the backward running filter produces another "measurement" to be incorporated into the forward running filter.

5.1.3 Discrete State Hidden Markov Models

Hidden Markov models (HMM) [Rabiner, 1989] with discrete states have the same graphical model and set of independence assumptions as a Kalman filter, and the standard inference algorithms use the same decoupling described in equations 5.5a and 5.5b. Also like a Kalman filter, an HMM model is specified by a transition model and an observation model. However, instead of integrating a continuous Gaussian density as in a Kalman filter, the inference algorithms involve a summation over discrete states.

As in the Kalman smoother, the distribution over a state variable X_k is found from a combination of a forward filter and a backward filter. Following convention, we define these filters as α and β so that the general smoothing equation 5.5b becomes

$$\begin{aligned} P(Y_{1:K}, X_k = x) &= \big[P(Y_{1:k-1}, X_k = x)P(Y_k \mid X_k = x)\big]\big[P(Y_{k+1:K} \mid X_k = x)\big] \\ &= P(Y_{1:k}, X_k = x)P(Y_{k+1:K} \mid X_k = x) \\ &= \alpha_k(x)\beta_k(x) \end{aligned} \tag{5.9}$$

The recursive formulas for the forward-backward computations $\alpha_k(x)$ and $\beta_k(x)$ given below follow the notational conventions in [Rabiner, 1989], with observation model $b_x(Y_k) = P(Y_k \mid X_k = x)$ and transition matrix $A(x', x) = P(X_k = x \mid X_{k-1} = x')$ giving the probability of transitioning from state x' to x.

$$\begin{aligned} \alpha_k(x) &= P(X_k = x, Y_{1:k}) \\ &= P(Y_k \mid X_k = x, Y_{1:k-1})P(X_k = x, Y_{1:k-1}) \\ &= P(Y_k \mid X_k = x)\sum_{x'} P(X_k = x \mid X_{k-1} = x')P(X_{k-1} = x', Y_{1:k-1}) \\ &= b_x(Y_k)\sum_{x'} A(x', x)\alpha_{k-1}(x') \end{aligned} \tag{5.10}$$

$$\begin{aligned} \beta_k(x) &= P(Y_{k+1:K} \mid X_k = x) \\ &= \sum_{x'} P(X_{k+1} = x' \mid X_k = x)P(Y_{k+1} \mid X_{k+1} = x')P(Y_{k+2:K} \mid X_{k+1} = x') \\ &= \sum_{x'} A(x, x')b_{x'}(Y_{k+1})\beta_{k+1}(x') \end{aligned} \tag{5.11}$$

The forward filter $\alpha_k(x)$ gives the joint probability of a specific state x and all the observations up to k. The recursive step involves summing over all the possible state

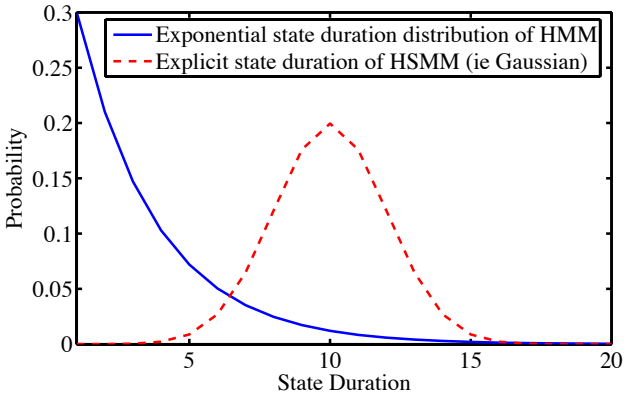


Figure 5.3: The prior distribution over the state duration in an HMM is exponentially decreasing, whereas the prior distribution for an HSMM can be modeled explicitly

transitions and then including the current observation. This is similar to the Kalman filter: the state is *predicted* forward, and then the state is *updated* with the current observation.

The backward filter $\beta_k(x)$ gives the probability of the future observations given the current state x. The recursive step starts with the probability of the observations a step in the future, includes the next observation, and then sums over the possible transitions away from the current state. This is similar to the backward running filter in the Kalman smoother.

5.1.4 Discrete State Hidden Semi-Markov Models

Conventional HMMs apply the state transition matrix A at every step, so that staying in a single state x for multiple steps requires repeated self-transitions. This results in a geometric distribution on the duration of each state, as shown in Figure 5.3. In many applications, an exponentially decreasing distribution is not appropriate because one may have prior information about how long the system tends to stay in a particular state. A hidden semi-Markov model (HSMM) (or segment model) [Ostendorf et al., 1996] [Rabiner, 1989] [Murphy, 2002] includes an explicit prior distribution over the expected duration of each state, so it can handle duration priors such as the truncated Gaussian shown in Figure 5.3.

Figure 5.4 gives the intuition for a hidden semi-Markov model. Unlike an HMM which transitions (possibly to the same state) at every step, an HSMM stays in a single state for some duration H, generating observations from that state at each step. Then, it changes state according to the transition matrix A and remains in the next state for some duration, generating observations from that state. The transitions between states remain Markov, but the individual steps are not Markov since the probability of transitioning depends on how long the system has been in that state, resulting in the name hidden *semi*-Markov model.

HSMMs use a variant of the standard forward-backward dynamic programming solution described in section 5.1.3 for inference in regular HMMs. The forward-backward compu-

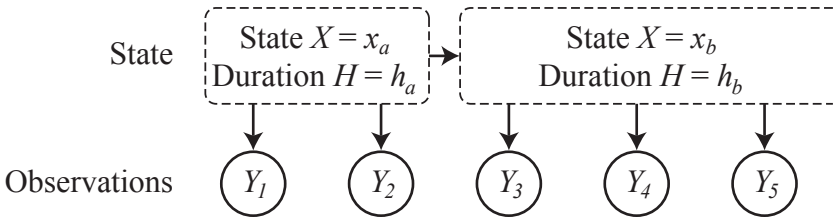


Figure 5.4: Intuition for a hidden semi-Markov model

tations for an HSMM [Ostendorf et al., 1996] [Rabiner, 1989] are still performed over the individual spatial steps X_k like in an HMM, but an HSMM must solve for the duration of each state, so in addition to summing over possible state transitions x' like in an HMM, we also sum over possible state durations h (we use the variable h for duration because duration represents *height* in subsequent chapters).

The following equations include the state transition matrix $A(x', x) = P(x \mid x')$, the observation model $b_x(Y_k^{k+h}) = P(Y_{k:k+h} \mid x, h)$ that gives the probability of a sequence of observations for a given state, and the explicit state duration prior $P(h|x)$ (see example in Figure 5.3). The α and β equations for an HMM in 5.10 and 5.11 give the probability of $X_k = x$. For an HSMM, the state x may be active over a number of spatial steps along the chain, so we instead find the probability that state x *ends* at step k.

$$\begin{aligned}
\alpha_k(x) &= P(\text{state } x \text{ ends at } k, Y_{1:k}) \\
&= \sum_{x'} \sum_{h} P(X_k = x, X_{k-h} = x', H_k = h, Y_{1:k}) \\
&= \sum_{x'} \sum_{h} P(Y_{k-h+1:k} \mid x, h) P(h \mid x) P(x \mid x') P(\text{state } x' \text{ ends at } k-h, Y_{1:k-h}) \\
&= \sum_{x'} \sum_{h} b_x(Y_{k-h+1}^{k}) P(h \mid x) A(x', x) \alpha_{k-h}(x')
\end{aligned} \tag{5.12}$$

$$\begin{aligned}
\beta_k(x) &= P(Y_{k+1:K} \mid \text{state } x \text{ ends at } k) \\
&= \sum_{x'} \sum_{h} P(Y_{k+1:K} | X_k = x, X_{k+h} = x', H_{k+} = h) \\
&= \sum_{x'} \sum_{h} P(Y_{k+1:k+h} \mid x', h) P(h \mid x') P(x' \mid x) P(Y_{k+h+1:K} \mid \text{state } x' \text{ ends at } k+h) \\
&= \sum_{x'} \sum_{h} b_{x'}(Y_{k+1}^{k+h}) P(h \mid x') A(x, x') \beta_{k+h}(x')
\end{aligned} \tag{5.13}$$

For many problems, the observations are conditionally independent given the current state, so the observation model becomes

$$b_x(Y_k^{k+h}) = P(Y_{k:k+h} \mid x, h) = \prod_{k'=k}^{k+h} P(Y_{k'} \mid x) \tag{5.14}$$

If the transition structure $A(x, x')$ is a deterministic chain, so that each state has only one possible transition, then equations 5.12 and 5.13 can be further simplified. We no longer need to sum over x' since the transition matrix A is 1 for the deterministic transitions and 0 for all other possible states. We use the notation x^- and x^+ to refer to the previous and next states in the deterministic chain transition structure. The following equations for α_k^{chain} and β_k^{chain} include this deterministic transition simplification as well as the conditional independence of observations in equation 5.14.

$$\alpha_k^{chain}(x) = \sum_h \left[\prod_{k'=k-h+1}^{k} P(Y_{k'} \mid x) \right] P(h \mid x) \alpha_{k-h}(x^-) \tag{5.15}$$

$$\beta_k^{chain}(x) = \sum_h \left[\prod_{k'=k+1}^{k+h} P(Y_{k'} \mid x^+) \right] P(h \mid x^+) \beta_{k+h}(x^+) \tag{5.16}$$

5.2 Markov Random Fields

The Markov random field (MRF) [Geman and Geman, 1984] [Besag, 1986] [Li, 2001] is an extension of the one dimensional Markov chain models described above to two (or more) dimensions, and it is commonly used to model structure in spatial domains. It has been successfully applied to many applications in computer vision including segmentation, noise reduction, surface reconstruction, and texture classification [Li, 2001], as well as other fields such as statistical mechanics [Swendsen et al., 1992].

As shown in Figure 5.5, a two-dimensional MRF consists of an undirected graph (often with a lattice structure) of nodes X_{ij}. Each node X_{ij} within the set of all nodes X_S is connected to a set of neighbors $X_{N_{ij}}$, describing the Markov property of the field

$$P(X_{ij} \mid X_S) = P(X_{ij} \mid X_{N_{ij}}) \tag{5.17}$$

The structure in a Markov random field acts as a prior $P(X)$ on the model that encodes smoothness, class continuity, or other properties. To estimate the state X from data Y, we use Bayes' rule

$$P(X \mid Y) \propto P(Y \mid X) P(X) \tag{5.18}$$

As in hidden Markov models, each state node X_{ij} produces a measurement Y_{ij}, and the measurements are conditionally independent given the state

$$P(Y_{ij} \mid X_S, Y_S) = P(Y_{ij} \mid X_{ij}) \tag{5.19}$$

This allows us to factor $P(Y \mid X)$ from equation 5.18

$$P(Y \mid X) = \prod_{ij} P(Y_{ij} \mid X_{ij}) \tag{5.20}$$

It is natural to specify a Markov random field in terms of its conditional distributions that characterize its local Markov relationships. For example, the 4-neighborhood system in

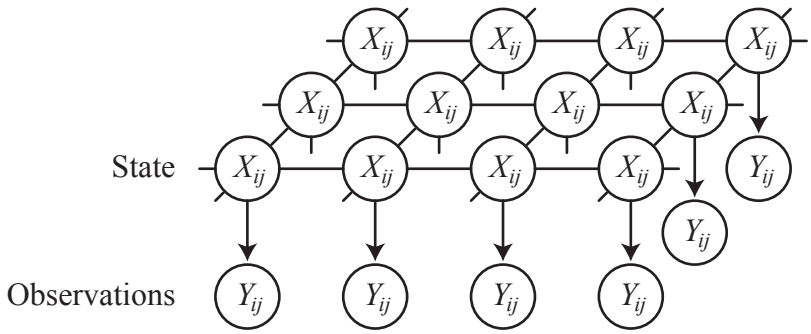


Figure 5.5: Graphical model of a Markov random field (all the state nodes X_{ij} have a corresponding observation node Y_{ij}, but some are not shown to make the figure readable)

the MRF model in Figure 5.5 contains only pairwise neighbors, so we can define *potentials* on single nodes V_1 and pairs of nodes V_2 and write the conditional distribution as

$$P(X_{ij} \mid X_{N_{ij}}) = \frac{\exp\left(-\left[V_1(X_{ij}) + \sum_{st \in N_{ij}} V_2(X_{ij}, X_{st})\right]\right)}{\sum_{X_{ij} \in \mathcal{L}} \exp\left(-\left[V_1(X_{ij}) + \sum_{st \in N_{ij}} V_2(X_{ij}, X_{st})\right]\right)} \tag{5.21}$$

where the normalizing constant is summed over all possible values $\mathcal{L}$ of X_{ij}.

Although the local characteristics in equation 5.21 are what we desire when we specify an MRF model, we need to specify the joint probability $P(X)$ over the entire field in order to perform inference and learn parameters. Let us define an *energy function* $U(X)$ as the sum over potential functions of sets of neighbors. Continuing with the 4-neighborhood example, we have

$$U(X) = \sum_{ij \in S} V_1(X_{ij}) + \sum_{ij \in S} \sum_{st \in N_{ij}} V_2(X_{ij}, X_{st}) \tag{5.22}$$

Then the *Hammersley-Clifford Theorem* [Li, 2001] [Winkler, 2003] states that the joint probability is a Gibbs field

$$P(X) = \frac{1}{Z} \exp(-U(X)) \tag{5.23}$$

where Z is a normalizing constant called the *partition function*.

Unfortunately, the partition function Z is a sum over all possible configurations of X, which makes it intractable except in special cases (such as the Gaussian MRF described in section 5.2.3). For example, classifying a 16 by 16 image into 4 classes would require a sum over 4^{16*16} configurations to evaluate the partition function.

$$Z = \sum_{X} \exp(-U(X)) \tag{5.24}$$

Specifying an MRF requires giving the form and parameters of the potential functions V. The following sections describe how learning and inference are done in an MRF, and then examples are given for a continuous state MRF and a discrete state MRF.

5.2.1 Learning

We consider the problem of learning from known labeled data. The problem is more involved if the parameters must be learned from noisy measurements only [Li, 2001].

Maximum Likelihood

A natural approach to learning the parameters θ of an MRF from labeled training data X is to find the maximum likelihood estimate

$$\begin{aligned} \theta^* &= \arg\max_\theta P(X \mid \theta) \\ &= \arg\max_\theta \frac{1}{Z(\theta)} \exp(-U(X \mid \theta)) \end{aligned} \tag{5.25}$$

However, this requires evaluating the partition function which is generally intractable, so approximate techniques that are based on the conditional probabilities are often used.

Pseudo-Likelihood

Pseudo-likelihood approximates the true likelihood with the product of the conditional likelihoods

$$PL(X) = \prod_{ij \in S} P(X_{ij} \mid X_{N_{ij}}) \tag{5.26}$$

The conditional likelihoods (equation 5.21) do not involve the partition function and can be maximized, but since X_{ij} and $X_{N_{ij}}$ are not independent, the pseudo-likelihood is not a true likelihood. However, in the large lattice limit, the maximum pseudo-likelihood is consistent if the model is correct [Winkler, 2003].

There are many other approximations that rely on the local properties of the field. See [Li, 2001] for a survey.

5.2.2 Inference

Once the model parameters are known, it can be used as a prior to help constrain ambiguous or noisy data. As in the hidden Markov models of section 5.1.3, we use a transition model $P(X)$ and an observation model $P(Y \mid X)$ to find an estimate of the state X

$$P(X \mid Y) \propto P(Y \mid X)P(X) \tag{5.27}$$

MAP

The maximum *a posteriori* (MAP) estimate is found by optimizing the posterior distribution from equation 5.27

$$\begin{aligned} X^* &= \arg\max_X P(X \mid Y) \\ &= \arg\max_X P(Y \mid X)P(X) \end{aligned} \tag{5.28}$$

If the observation model can be written in terms of an energy

$$P(Y \mid X) \propto \exp(-U(Y \mid X)) \tag{5.29}$$

then this can be combined with the Gibbs field $P(X)$ from equation 5.23 to rewrite the MAP estimate in terms of the log posterior

$$X^* = \arg\min_X U(Y \mid X)U(X) \tag{5.30}$$

Equation 5.30 does not involve the partition function since it is just a normalizing constant, and there are many global optimization techniques available, such as simulated annealing, that can be used to optimize this function. See [Li, 2001] for a survey.

Gibbs Sampling

As an alternative to global optimization, sampling methods [Robert and Casella, 2004] [Winkler, 2003] can be used to generate samples from the posterior distribution $P(X \mid Y)$. Expectations and variances can then be calculated from these samples to find an estimate.

The Gibbs sampler [Geman and Geman, 1984] is particularly well suited for sampling from an MRF because it produces samples based on conditional probabilities. Given a current configuration, the Gibbs sampler holds all the nodes fixed except for one, and then it generates a sample for that node from its conditional distribution.

$$\begin{aligned} &\text{sample } X_{11}^{'} \text{ from } P(X_{11} \mid X_{N_{11}}) \\ &\text{sample } X_{12}^{'} \text{ from } P(X_{12} \mid X_{N_{12}}) \\ &\ldots \end{aligned} \tag{5.31}$$

This process is continued across all the nodes and then repeated. In the limit, the samples generated by this process converge to the true distribution [Robert and Casella, 2004]. However, sampling methods often have high computational requirements because the conditional distributions must be sampled many times, and it is generally difficult to show that the sampling process has converged to the true distribution.

Minimizing posterior loss results in the MAP estimate and can be found using Gibbs sampling along with simulated annealing [Geman and Geman, 1984]. Minimizing the number of misclassified nodes in a discrete classification problem results in the marginal posterior mode (MPM)

$$X_{ij}^* = \arg\max_{X_{ij}} P(X_{ij} \mid Y) \tag{5.32}$$

and is found by choosing the most frequently sampled label at each node after convergence [Melas and Wilson, 2002].

5.2.3 Continuous State

A common MRF with continous state is the Gaussian Markov Random Field (GMRF) [Kashyap, 1981] [Lakshmanan and Derin, 1993] [Bouman and Sauer, 1993]. A GMRF has a Gaussian conditional distribution that makes it appropriate for smooth surfaces

$$P(X_{ij} \mid X_{N_{ij}}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2} \Big(X_{ij} - \sum_{\{s,t\} \in N_{ij}} \beta X_{st} \Big)^2 \right) \tag{5.33}$$

If we arrange the array of variables X_{ij} by rows into a single column vector X of size $M = rows * cols$, then we can write the joint probability of a GMRF as
[Lakshmanan and Derin, 1993] [Krishnamachari and Chellappa, 1997]

$$P(X) = \frac{1}{(2\pi\sigma^2)^{M/2}\sqrt{\det(\Sigma)}} \exp\left(-\frac{1}{2\sigma^2} X^T \Sigma^{-1} X\right) \tag{5.34}$$

where the inverse covariance Σ^{-1} is a sparse matrix with 1's on the diagonal, $-\beta$ at locations s, t whenever X_s and X_t are neighbors, and zeros everywhere else.

Equation 5.34 shows that unlike in the general case, the normalizing partition function of a GMRF takes on a simple form. By further assuming the field is isotropic and by choosing $\beta = 1/|N|$ where $|N|$ is the size of the local neighborhood (i.e. $\beta = 1/4$ in the 4-connected case), the exponent in equation 5.34 simplifies to a sum over pairs of neighboring nodes [Bouman and Sauer, 1993]

$$P(X \mid \sigma) = \frac{1}{(2\pi\sigma^2)^{M/2}\sqrt{\det(\Sigma)}} \exp\left(-\frac{1}{2\sigma^2} \sum_{ij,st\in\mathcal{N}} \frac{1}{|N|}(X_{ij} - X_{st})^2\right) \tag{5.35}$$

where $\mathcal{N}$ is the set of all neighboring nodes. Differentiating the log of the likelihood in equation 5.35 with respect to σ and setting the result to zero gives a simple analytic equation for the ML estimate of σ in terms of pairs of neighboring nodes [Saquib et al., 1998]

$$\hat{\sigma}^2_{\text{GMRF}} = \frac{1}{M|N|} \sum_{ij,st\in\mathcal{N}} (X_{ij} - X_{st})^2 \tag{5.36}$$

Since $M|N|$ is the number of pairs of neighbors in $\mathcal{N}$ that the sum is over, equation 5.36 is just the variance of pairwise differences. The simplicity of this result shows the benefit of using a GMRF. Finding the ML parameters for an MRF is intractable in general because of the partition function Z, but in the GMRF case when true realizations X of the field are known (uncorrupted by measurement noise), the ML parameter estimate is a simple summation over all the pairs of nodes. If a true realization X is not available and one only has noisy measurements Y, then EM can be used to estimate the parameters [Saquib et al., 1998].

Figure 5.6 shows a GMRF example of surface recovery from noisy data. The GMRF was trained using noisy data Y with the true surface X known. An independent Gaussian noise model was assumed, and the Markov assumption in equation 5.20 was then used to find the maximum likelihood estimate of the noise standard deviation $\hat{\sigma}_{\text{noise}} = 0.3$. A realization of the true surface X was used to train the GMRF prior according to equation 5.36, and this resulted in the maximum likelihood parameter $\hat{\sigma}_{\text{GMRF}} = 0.165$. Figure 5.6 shows that despite the large amounts of noise, the GMRF was able to recover the true surface.

The previous example had perfect training data so it was able to recover the true model parameters. Figure 5.7 shows the GMRF reconstruction with incorrect parameters $\hat{\sigma}_{\text{noise}} = 0.5$ and $\hat{\sigma}_{\text{GMRF}} = 0.1$. Increasing the assumed measurement noise and reducing the expected variation in the GMRF prior both have an increased smoothing effect on the result, which is evident in Figure 5.7.

As discussed in the results of chapter 4, a smoothing prior may not be appropriate if the desired output includes a tall obstacle such as a person. Figure 5.8 shows that a

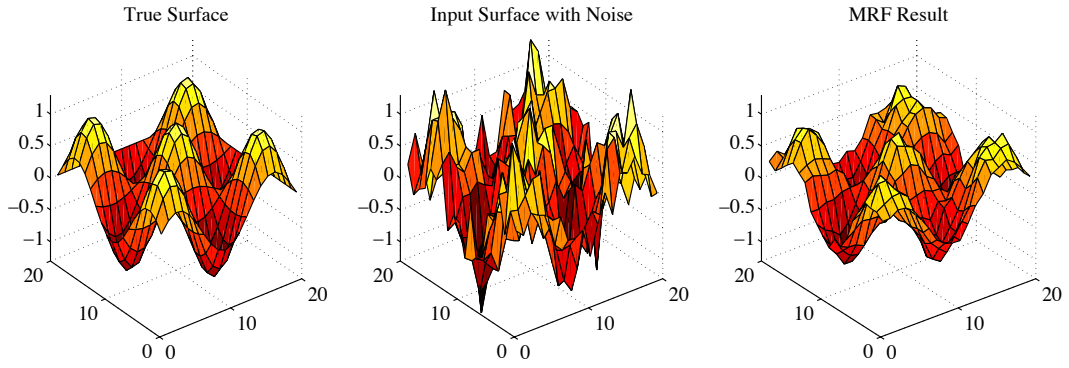


Figure 5.6: GMRF example with correct parameters

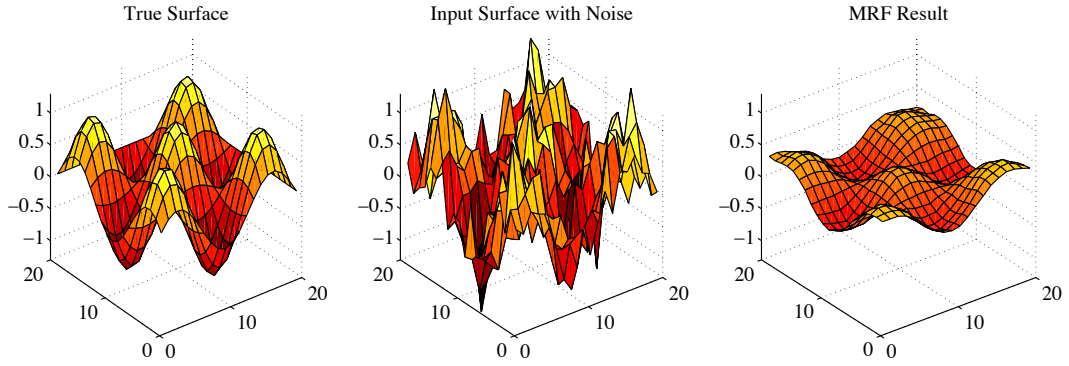


Figure 5.7: GMRF example with incorrect parameters

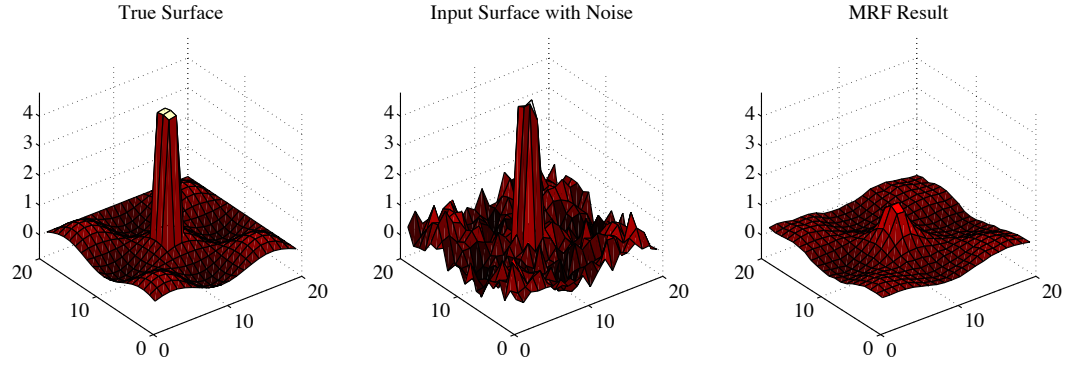


Figure 5.8: GMRF over-smoothing an obstacle

GMRF smoothes away an obstacle. Edge-preserving filters such as a median filter could be used in this case, but a more elegant solution may be to segment the ground and obstacle and treat them separately. This will be done in chapter 6, but first we will describe MRF segmentation into a set of discrete classes.

5.2.4 Discrete State

The simplest discrete MRF has a binary state and is known as the *auto-logistic* or *Ising* model [Geman and Geman, 1984]. This can be generalized to handle more than two states, and is then called the *multi-level logistic* (MLL) model [Li, 2001]. This model penalizes transitions so it is often used in segmentation problems to find blob-like regions.

The conditional distribution of the MLL follows equation 5.21. If we assume no single node potentials, then the model is fully specified by the pair-wise potential

$$V_2(X_{ij}, X_{st}) = \lambda(X_{ij} \neq X_{st}) \tag{5.37}$$

giving a conditional distribution

$$P(X_{ij} \mid X_{N_{ij}}) = \frac{\exp\left(-\lambda \sum_{\{s,t\} \in N_{ij}} (X_{ij} \neq X_{st})\right)}{\sum_{X_{ij} \in \mathcal{L}} \exp\left(-\lambda \sum_{\{s,t\} \in N_{ij}} (X_{ij} \neq X_{st})\right)} \tag{5.38}$$

which simply counts the number of neighbors that are different and multiplies the sum by the parameter λ which specifies the penalty for state transitions (a higher value of λ results in more smoothing).

The normalizing denominator for the conditional distribution in equation 5.38 is a simple sum over the possible values of X_{ij}. However, the normalizing partition function for the joint distribution (see equation 5.24) is intractable for this case because of the combinatorial number of possible configurations, so learning in this model requires approximate techniques such as pseudo-likelihood.

5.2.5 Extensions

Many extensions to MRFs have been proposed to handle various applications. The following methods are relevant to our modeling task.

Line Process

MRFs have been used extensively for texture segmentation, where the image is composed of a number of contiguous regions of constant texture. To enable this segmentation, a line process [Geman and Geman, 1984] has been used that acts as a break between neighboring textures. This idea could also be used to allow a discrete jump in an MRF surface representation to avoid smoothing the obstacle in Figure 5.8.

Double MRF

Another approach to handling texture segmentation is the double Markov random field [Melas and Wilson, 2002]. This approach uses one MRF layer to classify textures and another MRF layer that generates textures using separate parameters for each class. As above,

this could be used with a surface representation to allow discrete jumps at class boundaries. We take a related but different approach in chapter 6. As in a double MRF, we maintain a single MRF for class segmentation that interacts with another MRF representing the ground surface, but instead of changing the parameters of the ground MRF depending on the class, we interpret columns of data based on both the class MRF and the ground MRF.

3D MRF

Markov random fields have also been used to segment volumetric data stored in voxels. Many medical imaging devices produce 3D data, and MRFs have been used to segment various organs such as brains [Zhang et al., 2001]. A 3D MRF works the same as a 2D MRF, except that the neighborhood is larger and inference and learning require much more computation. Although we have a voxel representation to store our 3D data, our world model and our desired output consists of a classified ground surface with heights on it. As described in chapter 6, we exploit this structure by keeping the computationally demanding MRF structures 2D and handling our 3D data in terms of columns.

Chapter 6

Learning a Terrain Model with Spatial Dependencies

Most rough-terrain navigation approaches, including the online learning approach in chapters 3 and 4 make the strong assumption that individual patches of terrain are independent of each other. This chapter will show how the spatial Markov models presented in chapter 5 can be used to relax this assumption by including spatial correlations to model natural terrain and better interpret sensor data, especially when there is missing or ambiguous data. As described in chapter 1, the desired output is a local terrain model consisting of the supporting ground surface and the location of obstacles.

Section 6.1 describes the spatial correlations included in the model. The subsequent sections describe the data representation, the terrain model and how it encodes the desired spatial correlations, inference and learning in the model, run time considerations, and results. Chapter 7 compares this approach with the independent approach in chapter 4.

6.1 Spatial Correlation Assumptions

The assumption of independence between neighboring terrain patches can be justified because of simplicity or computational reasons, but it really does not describe actual terrain, especially in areas where a vehicle may drive. For example, the ground is generally smooth with a slowly changing height, so the ground heights of neighboring terrain patches are highly correlated. The ground also may contain occasional holes, ledges, and rock walls, but these are exceptional cases, and could be described as special types of obstacles in smooth ground.

Another natural assumption is that similar substances are near each other. Classes have continuity through space. If one patch of terrain is road, then the area around it is also likely to be road. Vegetation is the same way, especially in agricultural settings where an entire field has a uniform crop.

Vegetation of the same type is also likely to have a similar height. This is especially true in agricultural settings, but even in more general exploration domains, vegetation tends to have a common height. This assumption is different from the ground smoothness assumption in two ways. First, we expect more variation over vegetation height than ground height. Second, and more importantly, the ground smoothness assumption is for a

neighborhood, whereas the variation in vegetation height is over the entire area around the vehicle. A single stalk of vegetation may have a different height than its neighbor, but all of the stalks in a given area are likely to have a similar height.

Finally, for the domains we are considering, we expect a vertical orientation of substances because of gravity. We expect to see ground, then maybe some amount of substance (vegetation or obstacle), and then free space. This constraint could easily be expanded to handle overhanging obstacles such as tree branches, but by looking at the world as a set of columns, it will be difficult to find things like suspended wires whose only structure is horizontal. However, organizing the model in columns greatly reduces the computational requirements when compared to a full 3D model. Also, we believe that in our application domains the world basically consists of a ground surface with stuff on it, and we would like to exploit that knowledge in the model.

6.2 Data Representation

As described in section 1.2, our test platform is an automated John Deere 6410 tractor equipped with many sensors for localization and perception. The vehicle has a high-resolution stereo pair of digital cameras, an infrared camera, and two SICK laser range-finders (ladar) mounted on custom actively controlled scanning mounts. The first scanning ladar is mounted on the roof to get range data over a large area in front of the vehicle, and the second scanning ladar is mounted on the bumper to get high density measurements of nearby terrain and better penetrate upcoming vegetation. The cameras and scanned ladars are precisely calibrated and tightly synchronized with an accurate global vehicle pose estimate. Appendix A has more details about the sensors and positioning system.

The basic representational structure of our terrain model is the *voxel*: a 15cm^3 box-shaped region of 3 dimensional space. We represent the vehicle's spatial environment as a 3D voxel grid with indices (i, j, k), where $0 \leq i < I$, $0 \leq j < J$ and $0 \leq k < K$. We use k to index the vertical dimension, so the ijkth voxel is in the ijth position of a horizontal 2D grid and the kth position above an arbitrary subterranean origin.

Accurate global vehicle pose allows us to assign ladar points corresponding to the same region of space to the same voxel. Exploiting the precise synchronization of the sensors, we project ladar points into the most recent color and infrared images, so that each ladar point results in a vector of appearance measurements for that voxel, including laser remission (reflectance), infrared temperature, and color.[1]

The voxel representation also allows us to maintain a density estimate throughout space by comparing how many ladar rays pass through each voxel (pass-throughs) with the number of ladar rays that hit something in that voxel (hits). Density information is valuable when trying to separate sparse vegetation that contains a mixture of hits and pass-throughs from solid objects that contain a majority of hits and only a few pass-throughs due to sensor noise [Lacaze et al., 2002] (see Figure 1.4 in chapter 1).

[1]The ladar scans come in at a much higher rate than the image data so multiple scans are projected into the same image. However, the high pixel density of the images means that we collect approximately 100 pixels for every ladar point. This coupled with the continual movement of the scanning ladars makes it unlikely that a single pixel is used more than once, so we treat each color and infrared tagged ladar point as an independent measurement.

Although our data representation is based on the voxel, vehicle navigation is generally performed on a 2D surface, so our ground height estimates and classification results are made in terms of voxel columns. In our model, the ijth voxel column class is described with a multinomial distributed random variable C_{ij} taking on values related to the possible contents of the column, $C_{ij} = c$ with e.g. $c \in \{ground, vegetation, obstacle\}$.

Associated with the kth voxel in the ijth voxel column is the voxel state X_{ij}^k: a multinomial distributed random variable that describes the nature of the material inside the voxel, $X_{ij}^k \in \{ground, c, free\text{-}space\}$, where c is the class of the ijth voxel column.[2] The ijkth voxel is also associated with the observation vector $\mathbf{Y}_{ij}^k = [Y_{den}, Y_{rem}, Y_{ir}, Y_{col}]$, containing vectors of N ladar hit and pass-through density measurements of which the M hits include laser remission values, infrared temperatures and color data (i.e. $Y_{den} = [Y_{den}^1, \ldots, Y_{den}^N]$, $Y_{rem} = [Y_{rem}^1, \ldots, Y_{rem}^M]$).

6.3 Terrain Model

We use a probabilistic generative model to encode the desired spatial correlation assumptions described in section 6.1. The following sections describe the model in detail at three different levels:

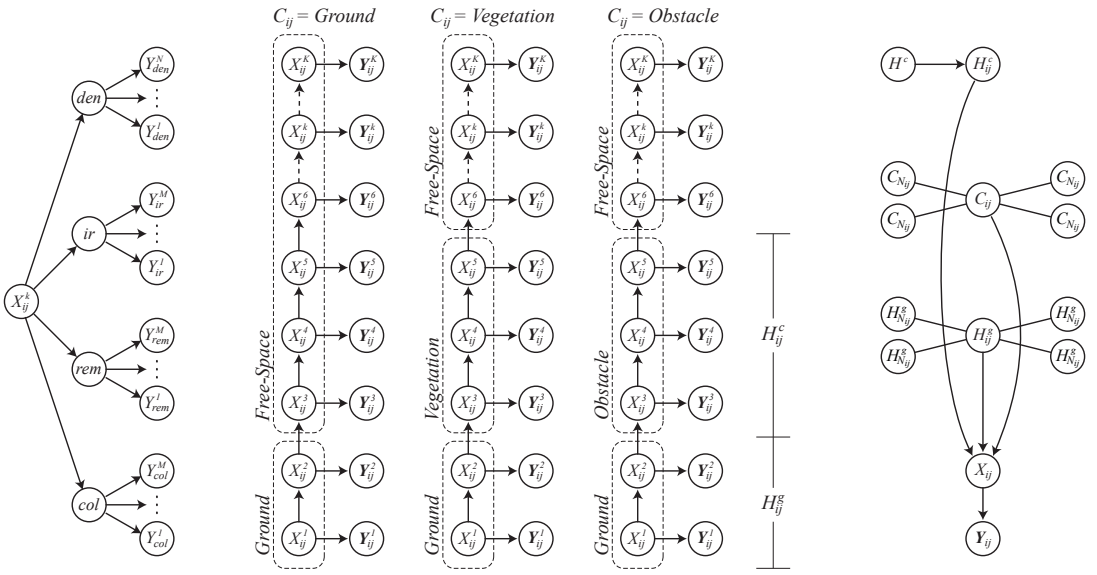
- Voxel observation models
- Vertical column structure
- Horizontal neighborhood structure

These sections build on each other from the inside out, going from an individual voxel to a voxel column to the entire grid of voxel columns. Figure 6.1 gives a graphical model representation of each of these levels and makes the relationships between the different variables explicit.

Before diving into the details of the model, we provide some context by exploring the generative nature of the model. We describe the process of generating data from our model, starting from the outside and working our way in. Figure 6.1(c) gives the neighborhood structure that encodes our three main assumptions from section 6.1: class continuity, smooth ground, and similar vegetation height. Figure 6.1(c) shows that the states X_{ij} and observations $\mathbf{Y}_{ij}$ in a voxel column ij are dependent on the class C_{ij}, ground height H_{ij}^g, and class height H_{ij}^c. These three quantities are dependent on their neighbors N_{ij} and the common class heights H^c that store the typical vegetation heights for each class of vegetation. From a generative point of view, if we have the common class height and the classes and ground heights from the neighboring voxel columns, we can generate a class, ground height, and class height for this voxel column.

Figure 6.1(b) expands the state X_{ij} of a voxel column into the individual voxel states X_{ij}^k arranged vertically such that they obey our assumption from section 6.1 that each column contains *ground* states, then *class* states, and then *free-space* states. Our generated class from above determines what substance we transition to between *ground* and *free-space*. Our

[2] In our implementation, the possibility of a voxel column simultaneously containing obstacle and vegetation is excluded, though its inclusion is a simple extension of the model we present.



(a) Voxel model (b) Voxel column HSMM vertical structure (c) MRF neighbors

Figure 6.1: A graphical description of the model showing (a) the voxel, (b) the voxel column, and (c) the connections between voxel columns. For each voxel column ij, the model contains voxel states X^k_{ij}, observations Y^k_{ij}, and a class C_{ij}, class height H^c_{ij}, and ground height H^g_{ij} that interact with neighbors N_{ij}. The common class heights over the entire local field are stored in H^c.

generated ground height and class height determine the *durations* of each state before it transitions to the next state.[3]

Figure 6.1(a) shows a single voxel state X^k_{ij} with its associated material properties and observations. For each voxel state from our generated voxel column from above, we can generate a set of material properties (e.g. density *den*) and from each of these material properties, we can generate a set of observations (e.g. $[Y^1_{rem}, \ldots, Y^M_{rem}]$).

This section has described our model from a generative point of view, including how the model can generate observation data from model variables. In general, we are interested in the opposite problem, when we have real sensor data and use a probabilistic inference algorithm (see section 6.4) to find the distribution over the class, ground height, and class height model variables given the data and the model structure.

The following sections describe the three levels of the model in more detail.

6.3.1 Voxel Observation Models

We assume that voxels form the smallest indistinguishable element of space, occupied completely by one (and only one) voxel state. Each voxel state maintains a distribution over

[3]We borrow the word *duration* from the HMM/HSMM literature (often used for time-series modeling) to describe the number of voxels of a particular substance before a transition to a different substance.

material properties including density, remission, infrared temperature, and color that describe the characteristics of that state, but the material inside a single voxel is assumed to be uniform. For example, the vegetation state may include a range of colors, and therefore different voxels in vegetation may have different colors, but we assume that the color of the vegetation within each voxel is uniform.

The measurement vector, $\mathbf{Y}_{ij}^k$ contains a variable number of noisy measurements of the material properties. The graphical model in Figure 6.1(a) illustrates the conditional independencies between the voxel state X_{ij}^k, the material property random variables *den*, *rem*, *ir* and *col*, and the measurements. Conditional on X_{ij}^k, the material properties are independent, and conditional on the material properties, the measurements are independent.

The voxel material properties are not directly observed, and we are not concerned with their values beyond what they reveal about the state. Thus material properties constitute nuisance variables that we remove from the observation models through marginalization, as described below.

Appearance

The distributions over the voxel appearance properties, including infrared temperature, laser remission and color, are all inherently multi-modal and are therefore not well described by a simple parametric distribution. For example, remission values in vegetation are either high because of the strong reflectivity of chlorophyll, or very low due to small cross-sectional area. We resort to a mixture of Gaussians to describe the distribution of the material properties within a state.

Mixture models break up a complex density into a mixture of R simple densities

$$p(z) = \sum_{i=1}^{R} P(i)p(z \mid i) \tag{6.1}$$

where $P(i)$ represents the mixing coefficients (or mixing priors), and $p(z \mid i)$ are the mixture densities. For a Gaussian mixture model (GMM), each of the mixture densities is a Gaussian

$$p(z \mid i) = \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(-\frac{1}{2\sigma_i^2}\|z - \mu_i\|^2\right) \tag{6.2}$$

with mean μ_i and variance σ_i^2. The time complexity of a GMM is proportional to the number of mixtures R. The parameters of a GMM are generally found using expectation maximization.

If there is one Gaussian centered at every training point $\mu_i = z_i$ and all the mixtures have the same prior $P(i) = 1/R$ and the same variance $\sigma_i^2 = \sigma_b^2$, then GMM becomes kernel density estimation (KDE) with a Gaussian kernel. KDE only has one parameter, the bandwidth σ_b^2, and adding new training data is trivial, but its runtime is proportional to the size of its training data set, which makes it impractical for this problem because of our large training sets.

We develop the marginal distribution for the remission values, but the infrared and color data are determined analogously. The distribution of a material property such as remission

rem for a given state x is modeled with a GMM

$$\begin{aligned} p(rem \mid X_{ij}^k = x) &= \sum_{i=1}^{R} P(i) p(rem \mid X_{ij}^k = x) \\ &= \sum_{i=1}^{R} P(i) \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(-\frac{1}{2\sigma_i^2} \|rem - rem_i\|^2\right) \end{aligned} \tag{6.3}$$

where each mixture has mean rem_i and variance σ_i^2. Conditioned on the true remission of the voxel *rem*, the M measurements y_{rem}^q are independent and assumed Gaussian with measurement variance σ_y^2

$$\begin{aligned} p(y_{rem} \mid rem) &= \prod_{q=1}^{M} p(y_{rem}^q \mid rem) \\ &= \prod_{q=1}^{M} \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{1}{2\sigma_y^2} \|y_{rem}^q - rem\|^2\right) \end{aligned} \tag{6.4}$$

As described above, the true remission of the voxel *rem* is unimportant except for what it reveals about the state, so we integrate it out to get the distribution for the measurements in terms of the state

$$\begin{aligned} p(y_{rem} \mid X_{ij}^k = x) &= \int p(y_{rem}, rem \mid X_{ij}^k = x)\, d(rem) \\ &= \int p(y_{rem} \mid rem) p(rem \mid X_{ij}^k = x)\, d(rem) \end{aligned} \tag{6.5}$$

Plugging equations 6.3 and 6.4 into equation 6.5 and carrying out the integration, we get a mixture of Gaussians as a function of the mean of the measurements $\bar{y}_{rem}$, with variances based on the number of data points M, the measurement model variance σ_y^2, and the individual mixture variances σ_i^2

$$\begin{aligned} p(y_{rem} \mid X_{ij}^k = x) &= \int \prod_{q=1}^{M} p(y_{rem}^q \mid rem) \sum_{i=1}^{R} P(i) p(rem \mid rem_i)\, d(rem) \\ &= \sum_{i=1}^{R} P(i) \int \prod_{q=1}^{M} p(y_{rem}^q \mid rem) p(rem \mid rem_i)\, d(rem) \\ &= \sum_{i=1}^{R} P(i) \frac{1}{\sqrt{2\pi\left(\sigma_i^2 + \frac{\sigma_y^2}{M}\right)}} \exp\left(-\frac{1}{2\left(\sigma_b^2 + \frac{\sigma_y^2}{M}\right)} (\bar{y}_{rem} - rem_i)^2\right) \end{aligned} \tag{6.6}$$

Equation 6.6 shows that the marginal appearance distributions become more broad when there are few data points (M is small), reflecting the increased uncertainty in the material property and hence the state. Figures 6.2-6.5 show this effect for a set of remission training data from *ground* and *vegetation* classes. The obstacle class is uniform over the

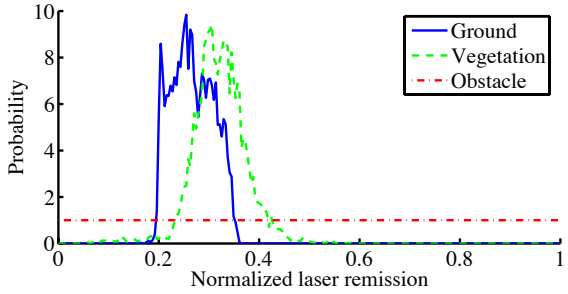


Figure 6.2: Laser remission training data

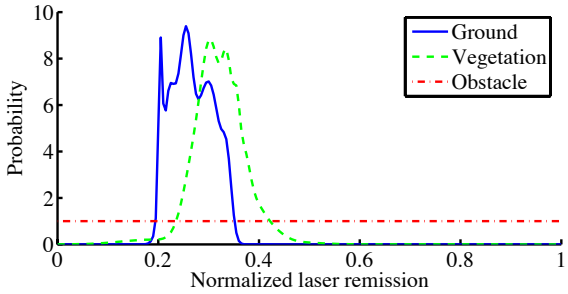


Figure 6.3: GMM remission observation model for an infinite number of measurements

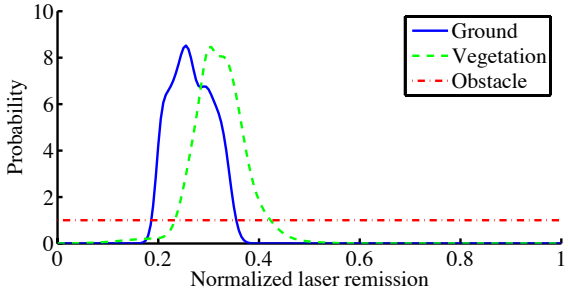


Figure 6.4: GMM remission observation model for 40 measurements

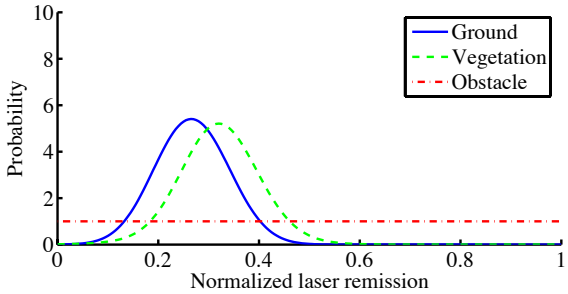


Figure 6.5: GMM remission observation model for 1 measurement

appearance data, which reflects our broad uncertainty about the characteristics of unknown obstacles. Figure 6.2 shows the raw data used for training the GMM observation models. For each class, a GMM with 10 mixtures was trained on this data using 100 steps of E-M. Figure 6.3 shows the observation model density that would be used if the voxel mean had been found using an infinite number of measurements. This density very closely matches the training data. Figure 6.4 shows the density if the voxel mean had been found with 40 measurements and displays some broadening. Figure 6.5 shows the large uncertainty in the observation models if the voxel only has a single measurement.

The use of the voxel material property variable (i.e. *rem* in Figure 6.1 and the above equations) that is then marginalized out is important. Without it, continued measurements of the same value for a given voxel would continually increase the probability of that voxel being a specific state. This is not correct, because continued measurements of the same value only make the system more certain about the actual material property *rem*, not the state x. Marginalizing the material property *rem* out, we get the desired behavior shown in Figures 6.2-6.5, where the densities converge to $p(rem \mid x)$ as more measurements are used.

Density

Voxel density values range from empty space ($den = 0$) to completely solid ($den = 1$). Analogously to the GMM models used for voxel appearance properties, the distribution of density values for a given state x can be well modeled using a beta distribution $B(a_x, b_x)$ which operates in the range $[0, 1]$ and has parameters a_x and b_x that together specify its mean and variance.

The measurements of density Y^n_{den} are binary (ladar hit or pass-through), so we use a binomial distribution to describe the number of hits $M = \sum_{n=1}^{N} Y^n_{den}$ out of N total rays for a given voxel density property *den*. As above, we integrate over the nuisance parameter *den*, and we recover the beta-binomial distribution as the marginal likelihood observation model.

$$\begin{aligned} P(M = m \mid X^k_{ij} = x) &= \int P(m \mid den) p(den \mid X^k_{ij} = x)\, d(den) \\ &= \binom{N}{M} \frac{B(a_x + M, b_x + N - M)}{B(a_x, b_x)} \end{aligned} \tag{6.7}$$

This model makes the assumption that a voxel with density *den* generates ladar hits that follow a binomial distribution (the outcome of repeated flips of a biased coin with $P(\text{heads}) = P(\text{hit}) = den$). However, since a given state x has a range of possible densities, which we model with a beta distribution, the distribution over hits M for a given state x becomes a beta-binomial, which is more broad than a binomial for low amounts of data N, but converges to a binomial as N becomes large.

Free-space

The *free-space* state does not possess any meaningful material properties beyond density *den*. Ladar hits occurring in *free-space* are generally the result of noise so we model the non-density material properties as matching the material properties of the states in contact with *free-space*. For example, the voxel above a *ground* state voxel may contain many pass-throughs with a single hit due to noise that has an appearance that matches the *ground*

state. If we modeled the appearance of *free-space* as uniform, then the strong match in appearance data with the *ground* state may overwhelm the density information and prevent the voxel from correctly being classified as *free-space*. By setting the appearance properties of *free-space* to match the state it is in contact with (*ground* in this example), the transition to *free-space* is decided solely on density information, which is appropriate since the *free-space* state does not possess any meaningful material properties beyond density.

Obstacles

Although we expect obstacles to generally have a fairly high density *den*, we cannot hope to build an accurate observation model for the appearance of each of the innumerable obstacles one might encounter in outdoor environments, so we simply use a single *obstacle* state with a corresponding uniform distribution over the observable range of material appearance properties (see Figure 6.4). We rely on accurately modeling the features of the trained states to detect obstacles as a default option when none of the other states are consistent. This is a common approach in anomaly detection, and has been used for detecting obstacles in an agricultural domain [Ollis and Stentz, 1997].

6.3.2 Vertical Column Structure

As discussed in section 6.1, when moving from lower to higher voxels within a column, we expect to move from ground to vegetation, or perhaps ground to obstacle, and eventually to free-space. We never expect free-space to be found below ground, nor do we expect vegetation to be suspended above free-space.

This type of structure is naturally imposed by introducing a Markov dependency between voxel states that restricts vertical transitions, thus defining a hidden Markov model within each voxel column. However, the duration of states such as *ground* and *vegetation* are not well modeled as states in a Markov chain which would induce a geometric distribution on the duration of states. We resort instead to a hidden semi-Markov model (HSMM) (see section 5.1.4) over voxel states, which explicitly represents a state duration (or height distribution) over voxels for each state value. These distributions over durations are provided from the neighborhood structure, as described in section 6.3.3.

As shown in figure 6.1(b), we associate a single HSMM chain structure with each column class C_{ij}, which makes the resulting column model a mixture of HSMMs. The durations of the ground and class states describe the height of those terrain elements and are given by H^g_{ij} and H^c_{ij}.

6.3.3 Horizontal Neighborhood Structure

The HSMM column models capture the vertical structure between the terrain elements or *states*, but there are also significant horizontal dependencies between neighboring columns, as discussed in section 6.1. As shown in Figure 6.1(c), we model these dependencies using two distinct but interacting Markov random fields (MRFs) for class C_{ij} and ground height H^g_{ij}, each dependent on the values of their respective neighbors, and a latent variable for the common class height H^c across all columns. These variables interact through the HSMM column models by imposing a prior on the state durations associated with H^c_{ij} and H^g_{ij} and imposing a prior over HSMM class models C_{ij}.

The neighborhood dependency of C_{ij} reflects the prior assumption that class identities are positively correlated with their neighbors so voxel columns tend to cluster in contiguous groups of the same class. We express this preference using the conditional MRF distribution (see section 5.2.4)

$$P(C_{ij} = c \mid C_{N_{ij}}) \propto \exp\Big(-\lambda_C \sum_{\{s,t\} \in N_{ij}} (c \neq c_{st})\Big) \tag{6.8}$$

where N_{ij} is the set of neighboring indices and $C_{N_{ij}}$ is the set of classes in the neighborhood of the ijth voxel column.

Ground height varies smoothly from one patch of ground to the next, so we expect that H^g_{ij} will be tightly correlated with nearby values. We express this belief using a Gaussian Markov random field (see section 5.2.3)

$$P(H^g_{ij} = h \mid H^g_{N_{ij}}) \propto \exp\Big(-\frac{1}{2\sigma_G^2}\Big(h - \frac{1}{|N_{ij}|} \sum_{\{s,t\} \in N_{ij}} h^g_{st}\Big)^2\Big) \tag{6.9}$$

where $|N_{ij}|$ is the size of the neighborhood.

We expect that vegetation of the same class c has a similar height H^c with some variation across the local field. This assumption may not be valid for obstacles, so we only apply it to vegetation classes. Given the common height of the vegetation in this area H^c, we model the expected variation with a Gaussian truncated by the interval of possible class heights $I_{[h^c_{min}, h^c_{max}]}$

$$P(H^c_{ij} = h \mid H^c) \propto I_{[h^c_{min}, h^c_{max}]} \exp\Big(-\frac{1}{2\sigma^2_{H^c}}(h - h^c)^2\Big) \tag{6.10}$$

6.4 Inference

The interacting Markov random fields of this model capture important structure, but these dependencies prevent analytic determination of the posterior distribution $P(C, H^g, H^c \mid Y)$. The set of HSMMs that describe the data in each column of voxels can efficiently produce distributions over the state durations, which makes it easy to sample from the conditional distribution

$$P(C_{ij}, H^g_{ij}, H^c_{ij} \mid Y_{ij}, C_{N_{ij}}, H^g_{N_{ij}}, H^c) \tag{6.11}$$

so we use Gibbs sampling (see section 5.2.2) for approximate inference over the MRFs.

Algorithm 1 gives the application of Gibbs sampling to our model. The HSMM column models require a distribution over class heights which comes from the common class height latent variable H^c, as shown in Figure 6.1(c) and described in section 6.3.3. Samples of the common class height are produced from its conditional distribution given the current column class height samples h^c_{ij}

$$P(H^c = h \mid H^c_{ij \in IJ}) \propto \exp\Big(\frac{-1}{2\sigma^2_{H^c}/D^c}\Big(h - \frac{1}{D^c} \sum_{ij \in IJ, c_{ij} = c} h^c_{ij}\Big)^2\Big) \tag{6.12}$$

where D^c is the number of columns with class c.

Algorithm 1 Gibbs sampling for inference in the terrain model

Sample common class heights h^c from $P(H^c \mid H^c_{ij \in IJ})$ using all the class height samples of the same class across the field of voxel columns
for all MRF voxel columns ij **do**
  Find ground and class priors from neighbors:
    $P(H^g_{ij} \mid H^g_{N_{ij}})$
    $P(C_{ij} \mid C_{N_{ij}})$
  for all Classes c **do**
    Find class height prior from common class height of same class:
      $P(H^c_{ij} \mid H^c)$
    Use class HSMM to find probability of the data and distributions over the ground and class height:
      $P(Y_{ij} \mid C_{ij} = c, H^g_{N_{ij}}, H^c)$
      $P(H^g_{ij} \mid C_{ij} = c, Y_{ij}, H^g_{N_{ij}}, H^c)$
      $P(H^c_{ij} \mid C_{ij} = c, Y_{ij}, H^g_{N_{ij}}, H^c)$
  end for
  Compute class distribution:
    $P(C_{ij} \mid Y_{ij}, C_{N_{ij}} H^g_{N_{ij}} H^c)$
      $\propto P(Y_{ij} \mid C_{ij}, H^g_{N_{ij}} H^c) P(C_{ij} \mid C_{N_{ij}})$
  Sample c_{ij} from $P(C_{ij} \mid Y_{ij}, C_{N_{ij}}, H^g_{N_{ij}}, H^c)$
  Sample h^g_{ij} from $P(H^g_{ij} \mid C_{ij} = c_{ij}, Y_{ij}, H^g_{N_{ij}}, H^c)$
  Sample h^c_{ij} from $P(H^c_{ij} \mid C_{ij} = c_{ij}, Y_{ij}, H^g_{N_{ij}}, H^c)$
end for

Once the common class heights across the local field H^c have been sampled, each voxel column is sampled. The first step of the sampling procedure is to find the priors over class C_{ij}, class height H^c_{ij} and ground height H^g_{ij} from the neighbors, as given in equations 6.8 and 6.9, and the common class heights H^c as given in equation 6.10. The priors on H^c_{ij} and H^g_{ij} are then incorporated into the HSMM model as priors over state durations and are shown in the subsequent equations as $P(H^c_{ij} = h \mid H^c)$ for the class state $x = c$ or $P(H^g_{ij} = h \mid H^g_{N_{ij}})$ for the ground state $x = g$.

Once the prior distributions are found, the class HSMM structures are used to find the probability of the data and the state duration probabilities for each class. HSMMs use a variant of the standard forward-backward dynamic programming solution used for inference in regular HMMs (see section 5.1.4). As shown in figure 6.1(b), an HSMM maintains durations (corresponding to height in our case) so that a single state is active over a number of spatial steps up the chain. This formalism is very natural for finding ground height or class height because the neighborhood information can be included as a prior on the corresponding state duration.

The forward-backward computations are still performed over the individual spatial steps X^k_{ij} as in an HMM, but an HSMM must solve for the duration of each state, so in addition to summing over possible state transitions x' like in an HMM, we also sum over possible state durations h. Equations 6.13 and 6.14 give the HSMM forward and backward probabilities $\alpha^k_{ij,c}$ and $\beta^k_{ij,c}$ for spatial step k of the class c chain in MRF voxel column ij. We take

advantage of the observation independencies and the deterministic transitions of our chain structures to simplify the following equations and reduce the computational complexity. We use the notation x^- and x^+ to refer to the previous and next states in the chain of the current class. See section 5.1.4 for more details (equations 5.15 and 5.16).

$$\begin{aligned}
&\alpha^k_{ij,c}(x) = P(\text{state } x \text{ ends at } k, Y^{1:k}_{ij} \mid C_{ij} = c, H^g_{N_{ij}}, H^c) \\
&= \sum_{x'} \sum_{h} P(X^k_{ij} = x, X^{k-h}_{ij} = x', H^k_{ij} = h, Y^{1:k}_{ij} \mid C_{ij}, H^g_{N_{ij}}, H^c) \\
&= \sum_{h} \prod_{k'=k-h+1}^{k} P(Y^{k'}_{ij} \mid x) P(H^x_{ij} = h \mid H^g_{N_{ij}}, H^c) \alpha^{k-h}_{ij,c}(x^-)
\end{aligned} \tag{6.13}$$

$$\begin{aligned}
&\beta^k_{ij,c}(x) = P(Y^{k+1:K}_{ij} \mid \text{state } x \text{ ends at } k, C_{ij} = c, H^g_{N_{ij}}, H^c) \\
&= \sum_{x'} \sum_{h} P(Y^{k+1:K}_{ij} | X^k_{ij} = x, X^{k+h}_{ij} = x', H^{x^+}_{ij} = h, C_{ij}, H^g_{N_{ij}}, H^c) \\
&= \sum_{h} \prod_{k'=k+1}^{k+h} P(Y^{k'}_{ij} \mid x^+) P(H^{x^+}_{ij} = h \mid H^g_{N_{ij}}, H^c) \beta^{k+h}_{ij,c}(x^+)
\end{aligned} \tag{6.14}$$

Since we know by assumption that the chain must end in the final state $x = \textit{free-space}$, the probability of the data for class c is the final value of α in that state.

$$P(Y_{ij} \mid C_{ij} = c, H^g_{N_{ij}}, H^c) = \alpha^K_{ij,c}(x = \textit{free-space}) \tag{6.15}$$

As described in Algorithm 1, this is combined with the class prior $P(C_{ij} \mid C_{N_{ij}})$ to find the distribution over classes, which is used to sample a new class.

Finding the distribution over state durations involves combining α and β. As above, equation 6.16 takes advantage of the deterministic transitions of the chain structures to reduce computation.

$$\begin{aligned}
&\zeta^x_{ij,c}(h) = P(\text{state } x \text{ has duration } h \mid Y_{ij}, C_{ij} = c, H^g_{N_{ij}}, H^c) \\
&= \sum_{k} P(X^k_{ij} = x, X^{k-h}_{ij} = x^- \mid Y_{ij}, C_{ij}, H^g_{N_{ij}}, H^c) \\
&= \sum_{k} \prod_{k'=k-h+1}^{k} P(Y^{k'}_{ij} | x) P(H^x_{ij} = h | H^g_{N_{ij}}, H^c) \alpha^{k-h}_{ij,c}(x^-) \beta^k_{ij,c}(x)
\end{aligned} \tag{6.16}$$

We know that in each chain, every state transition must occur after some duration, so we can normalize by $\sum_h \zeta^x_{ij,c}(h)$ to get the posterior on ground and class height conditional on the neighbors. Samples are then drawn from these distributions.

$$\begin{aligned}
P(H^g_{ij} = h \mid C_{ij} = c, Y_{ij}, H^g_{N_{ij}}, H^c) &= \zeta^{x=\textit{ground}}_{ij,c}(h) \\
P(H^c_{ij} = h \mid C_{ij} = c, Y_{ij}, H^g_{N_{ij}}, H^c) &= \zeta^{x=\textit{state } c}_{ij,c}(h)
\end{aligned} \tag{6.17}$$

The time complexity of HSMM calculations is greater than an HMM because of the sum over possible durations, but the observation likelihood products can be pre-computed

and the state durations to search over can be constrained based on the priors to reduce the complexity to $O(numVoxels * numStates * maxDuration)$ for a single chain.

Although it is typically difficult to show that Gibbs sampling has converged, we have found empirically that the model finds a good estimate quickly, allowing for real-time execution.

6.5 Learning

The model described in section 6.3 incorporates prior knowledge about the structure of the environment, but the specific model parameters must be learned from training data. These parameters include the sensor observation models for each state and the neighborhood interactions for class, class height, and ground height. The generative nature of our model allows us to decouple the learning problems, and train each of these observation and neighborhood interaction models individually, thus greatly simplifying the learning task.

6.5.1 Learning the Observation Models

Collecting labeled training data is often expensive, especially in outdoor environments where there can be high variation in sensor readings so that a large training set is needed. We use an approach based on the ideas in chapters 3 and 4 to collect large quantities of labeled training data to automatically train our observation models. Specifically, we drive through representative terrain of a single class such as *vegetation* and store the sensor measurements from the voxels of columns that we drive over as training examples for that class. This process is then repeated for other classes such as *ground*. Unlike the method presented in chapter 4 which directly trains on the height of different types of vegetation, this method only trains on the various material properties of vegetation voxels, allowing the system to remain general across vegetation heights.

Each labeled voxel collected by driving through representative terrain is used as a training example for the observation models in equations 6.6 and 6.7. For appearance data such as remission, infrared and color, the mean values from each voxel are used to train the GMM observation models (i.e. rem_i, σ_i^2, $P(i)$ in equation 6.6) and the variance of measurements within the voxels is used as the GMM measurement model variance (σ_y^2 in equation 6.6).

Hit and pass-through data from the labeled training voxels are used to find the maximum likelihood parameters of the beta-binomial density model (a_x and b_x in equation 6.7) for each class state x using a Newton-Raphson method[Smith, 1983]. This handles class states like *ground* and *vegetation*, but the density of *obstacle* and *free-space* states must also be trained. The *free-space* density can be trained using data that includes insects or dust that occasionally returns a ladar point, or it can just be set manually to strongly favor empty space. Similarly, the *obstacle* density can be trained using hit and pass-through data from representative obstacles, or it can be set manually to favor dense objects.

6.5.2 Learning the Neighborhood Models

The priors given in equations 6.8 and 6.9 describe how class and ground height depend on their neighbors, and the prior in equation 6.10 describes how column class heights are related to the common class height. Each of these priors contains a parameter that gives the

strength of the prior, and describes how much classes tend to clump together, how smooth the ground is, and how little class heights vary. As above, we train these parameters by driving over representative terrain.

As we drive over an area, we record the ground heights measured by the location of our wheels. We use these height sequences to find the standard deviation σ_G of typical ground height variation between voxel columns, which gives us the maximum likelihood estimate of our ground neighborhood prior. As described in section 5.2.3, the maximum likelihood estimate for a Gaussian MRF is simply the standard deviation of the pairwise differences.

Similarly, as we drive through vegetation, we get an approximate vegetation height measurement by taking the highest ladar hit and subtracting the known ground height (from the wheel locations). Since we assume that vegetation heights are independent given the common vegetation height in the area, we can find the class prior standard deviation σ_{H^c} directly from this sequence of class heights.

The class interaction prior λ_C gives the probability that a class transitions to a different class. This could be estimated directly using pseudo-likelihood methods (see section 5.2.4) with class-labeled data over a large area that includes many class transitions, but unlike the labeled data for the observation models or the ground and class height interactions, this type of training data is difficult to collect. However, changing the class interaction prior affects the system output in an intuitive way by controlling how much classes tend to clump together, so this parameter can be set manually.

6.6 Run Time

Performing inference in this model is intensive computationally because of the repeated calculations necessary to sample from the model. Our system runs a loop that updates the local terrain map at approximately 1Hz. Within this loop, the system computes the observation likelihood products, calculates 20 samples from each column in the terrain map, and updates the mean ground height, class height, and most likely class for each column.

At a vehicle speed of 1m/s, this results in approximately 200 samples for a terrain patch before the vehicle reaches it. Although sampling convergence is difficult to prove, the system generally finds the solution quite rapidly in our experiments, allowing us to run the system in real time.

6.7 Simulation Results

This chapter has presented a terrain model that includes spatial correlations to better handle missing and ambiguous data in dense non-penetrable vegetation without needing to explicitly train on the vegetation height. This section shows simulation results that verify these properties.

Figure 6.6 shows simulated data of a transition from ground to tall vegetation. Imagine the vehicle is approaching this from the left, so its sensors get measurements of the ground, the front of the vegetation, and the top of the vegetation, but since the vegetation is dense there are no range measurements of the ground surface under the vegetation. The dashed line gives the true ground height, showing that the ground under the vegetation is flat and then angles up a hill. The vegetation has some columns with missing data, and some

voxels in the vegetation match the appearance of ground, as shown by their dark blue color. Although there is no data under the ground or vegetation surfaces, the voxels above the ground and vegetation are full of pass-throughs. The ground and vegetation appearance models were set to match the simulated appearance data so we could study the effects of the spatial correlations in the model.

Since this example assumes the vehicle is approaching from the left, the system was initialized with the "ground" class, a ground height of 2, and a class height of 0 (the "ground" class has no height). The sampling inference procedure given in algorithm 1 was then run for 100 iterations (each iteration produces samples from every column) which took 0.5 seconds. The final 50 samples from each column were used to find the most common class, the mean ground height, and the mean class height (although we allowed 50 iterations of "burn in" time to let the sampling procedure converge, the system actually converged after approximately 5 iterations).

Figure 6.7 shows the ground height estimates and Figure 6.8 gives the class and class height estimates. These values represent the most likely explanation of the data given the prior knowledge encapsulated in the model. Although the system was never trained on the height of the vegetation, it was able to recover the vegetation height and use this to estimate the ground height including the hill. The visible ground surface along with the ground smoothness assumption constrains where the ground can be at the transition to vegetation. Therefore, the vegetation height at the transition is very likely and is propagated through the rest of the model because of the assumption of similar vegetation height.

The model structure also allowed the system to handle missing and ambiguous data. The class prior makes it likely that the missing data areas are vegetation, and the ground prior makes it likely that the ground is smooth in these areas. This interpretation is likely under the model and consistent with the data. The ambiguous data patches in the vegetation have appearance properties similar to ground, but the ground smoothness prior makes it extremely unlikely for the ground to be that high, so the model produces vegetation estimates in those areas. The structure in the model prior allowed the system to reject this unlikely data.

The class height estimates in Figure 6.8 are not completely uniform. There is a crease where the hill starts because the model ground prior enforces a smooth transition from the flat region to the hill in Figure 6.7 whereas the simulated data has an abrupt angle change. The class heights at the far right become slightly larger because of the conflict between the ground prior that wants a smooth flat ground and the class height prior that wants a uniform vegetation height.

The class height predictions are slightly lower in the missing data areas because of asymmetric data evidence. In the absence of any hits, the class prior would give the missing data areas a symmetric class height distribution around the true class height. However, the large number of pass-throughs above the missing data areas produces a strong constraint that cuts off the top of the class height prior distribution. Therefore the class height samples in areas with missing data are biased low. Since there are no hits in that patch, it is reasonable to expect that the vegetation height is lower in this area.

A system like the one presented in chapter 4 that assumes that terrain patches are independent would do poorly in this example. First, since the vegetation is dense and the ground is hidden, it would need to know how tall the vegetation is. The online learning system in chapter 4 could adjust to this height once it drove into it and could train on

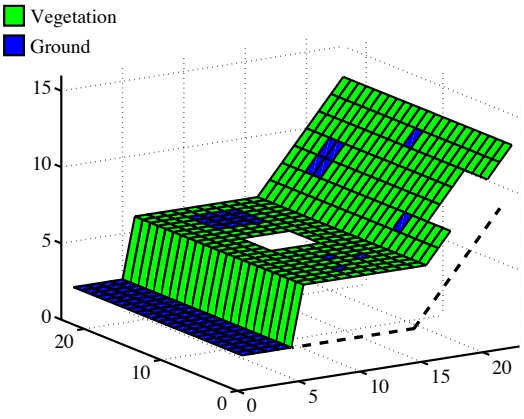


Figure 6.6: Simulated sensor data, showing transition from ground to tall vegetation with missing and ambiguous data. The dashed line gives the true ground height

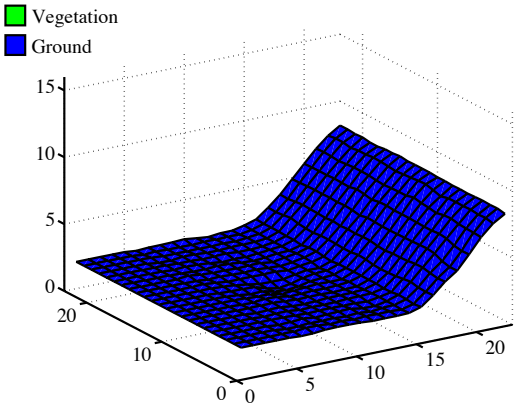


Figure 6.7: Ground height

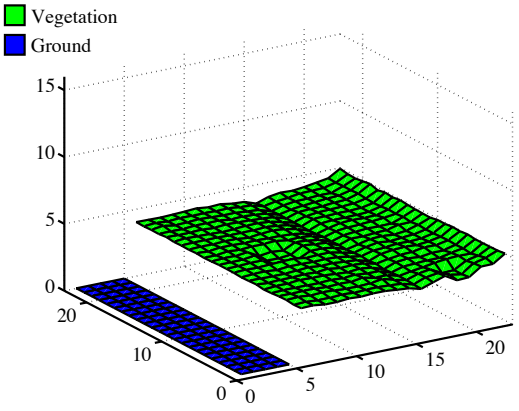


Figure 6.8: Class and class height

it, but it does not have the capability to use the weakly-labeled data at the ground to vegetation transition like this method. Also, a system that makes independent estimates would produce incorrect high ground estimates where there is ambiguous data, because it doesn't use the neighborhood to filter these out.

6.8 Results

We have tested this model in a nearby working farm and an undeveloped area with tall weeds. In each case, we train the system by driving it through representative terrain classes. The following examples show the benefits of including spatial correlations in areas with dense non-penetrable vegetation. We also show that the system can find an obstacle in vegetation and what happens when the ground smoothness assumption is broken.

The Gibbs sampling inference procedure described in section 6.4 produces samples from the posterior distributions on class, ground height, and class height. The results below use the most commonly sampled class and the mean of the ground height and class height samples (see section 5.2.2).

6.8.1 Comparison Algorithms

Current approaches that filter out vegetation from the ground surface generally rely on the deepest range penetration, but for dense non-penetrable vegetation, this performs very poorly since there are no range points that reach the ground. Therefore, in addition to comparing our model to the lowest hit or pass-through, we also compare it to an approach that adjusts the lowest point based on independent column classifications using the same sensor models but without any neighborhood interaction. We use the average height of each class from the training data for the offset. Additionally, chapter 7 presents a comparison between the model described in this chapter and the independent online learning approach described in chapter 4.

6.8.2 Transition to Dense Vegetation

Figure 6.9 shows a transition from low grass to tall weeds that is similar to the simulation example in section 6.7. We trained the system in similar low green grass and tall yellow weeds in a nearby area. This training procedure only recovers the voxel observation models and the neighborhood correlation constants, not an explicit vegetation height.

The colors in Figure 6.10 show the independent column classifications, and the heights in the figure show the lowest hit or pass for each column. This shows that the vegetation is dense and no range measurements penetrate beyond the first row of weeds.

Figures 6.11 and 6.12 show the system output, including ground heights, class heights, and class labels. The figures show that the class predictions are more continuous instead of the noisy class labels in Figure 6.10. More importantly, the model is able to infer the vegetation height from the transition and correctly predict the hidden ground surface.

Figures 6.13 and 6.14 compare the model ground height predictions with the lowest range point, the lowest point adjusted based on its independent classification, and the true height found when the vehicle drives into the tall vegetation. These figures show that the lowest point is a poor indicator of ground height in dense vegetation, but that the

Figure 6.9: View from tractor of transition from low grass to tall weeds

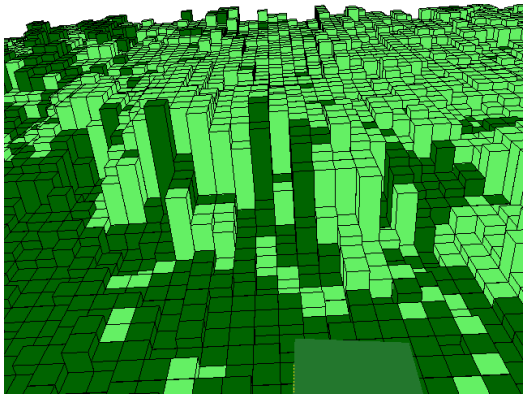


Figure 6.10: Lowest hit or pass with independent column classifications

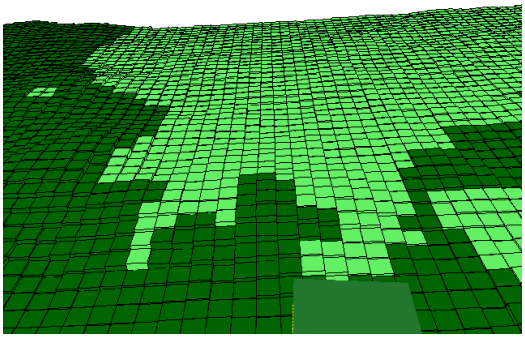


Figure 6.11: Ground height predictions

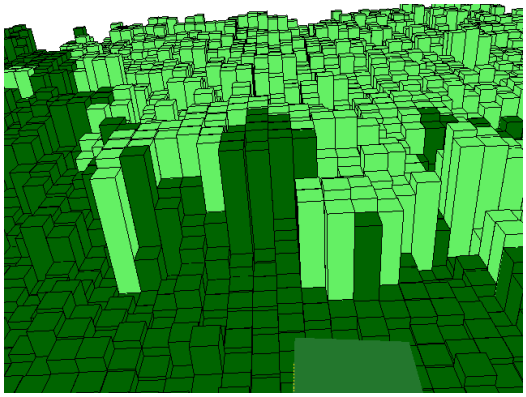


Figure 6.12: Class height predictions

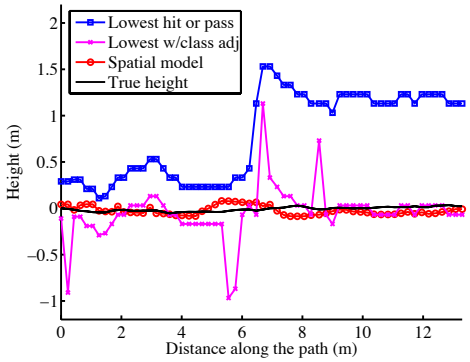


Figure 6.13: Ground height prediction comparison for left wheel

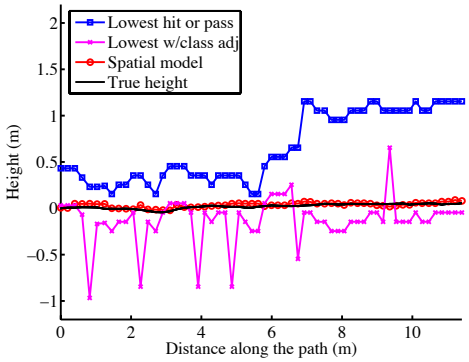


Figure 6.14: Ground height prediction comparison for right wheel

adjusted version does well when it is correctly classified. However, classifying each voxel columns independently without consideration of its neighborhood results in a number of misclassifications which significantly degrade the ground height estimates.

The spatial model also misclassifies some of the voxel columns at the transition because the side of the tall weeds can have an appearance more similar to short grass than the yellow tops of the tall weeds that the system was trained on. However, because of the ground smoothing prior, the effect of these misclassifications is small. The model ground height predictions show a small error at 5m in Figure 6.13 which corresponds to the misclassified columns in the center of Figure 6.12, but this error would not prevent successful navigation.

6.8.3 White Shed

Figure 6.15 shows the view from the tractor as it approaches a white shed. This is a large obstacle that could be reliably detected in a variety of ways, but it will serve as a good example of how the various pieces of our model interact to produce the correct result. Figure 6.16 shows the output of our model including the class labels: obstacle (red), vegetation (green), ground (gray), and the ground height for drivable areas. Obstacle columns are shown at their class height. The model produces a reasonable classification of the scene and a smooth ground estimate that would work well for vehicle navigation. It classifies the shed as an obstacle and correctly captures the hill sloping down to the right despite the presence of sparse vegetation.

This example is interesting because on a voxel basis the ground class is much more likely than the broad uniform obstacle class for the voxels from the shed. However, the MRF and HSMM spatial constraints imposed on the ground surface make it extremely unlikely that the ground height would have a tall step discontinuity at the shed wall. Since the density and appearance data are not well described by the vegetation class, the shed is correctly classified as an obstacle.

Figure 6.17 shows the output of the system when the neighborhood interactions are ignored and the columns are assumed to be independent. Without neighborhood information, classification is based solely on the data likelihood for each column HSMM model. Lacking the smooth ground prior, the wall is classified as a collection of tall columns of ground. A vision system that ignores 3D structure and only makes a classification based on the observation models we use would produce a similar result. Figure 6.17 also shows that without the ground and class priors, the ground height estimates and classification labels are generally more noisy.

6.8.4 Person in Vegetation

Figure 6.18 shows the view from the tractor in a challenging scene: a camouflaged person in tall vegetation with a small dirt mound to the right. Figure 6.19 shows that both the person and the dirt mound have a high temperature. A simple obstacle detection system that performed a threshold on temperature may classify both of these objects as obstacles. We show that the structure of our model allows the system to disambiguate these two objects.

We trained the model on ground, low grass, and tall weeds as in section 6.8.2. Figure 6.20 gives the ground heights and classification results. The model correctly classified the

Figure 6.15: View from the tractor of a white shed

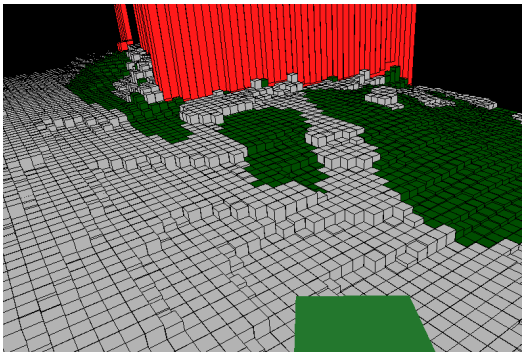


Figure 6.16: System output, including ground heights and classification

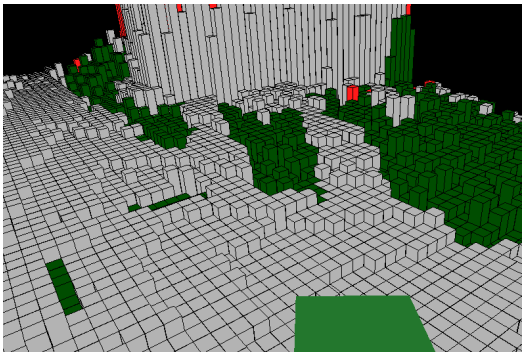


Figure 6.17: System output with neighborhood interactions turned off, showing incorrect classification of the shed

person and the dirt mound as well as the two types of vegetation. The area in the shadow of the vegetation is classified as ground. Although that area is actually low grass, ground is a reasonable guess since the system has no data in those columns except for pass-throughs high above the ground height.

The use of both the known ground height under the vehicle and the model structure allowed the model to produce reasonable estimates of the ground height even in areas where the ground is hidden. In addition to providing a smoothing prior, neighborhood interactions allow information to propagate. Fixing the heights under the wheels affects the ground estimates in the surrounding area. Columns with little or no data can still produce useful estimates using their neighborhood. Class height similarity helps infer the ground height in areas where the ground is not directly observable. Knowing the ground height allowed the model to explain the dirt mound as a rise in the ground. Knowing the ground height similarly allowed the model to reject the person being classified as ground, so the person was classified as an obstacle.

Figure 6.21 shows the predicted class heights, giving the tops of the vegetation. As in section 6.8.2, the sides of the tall weeds are classified as low grass, which causes a small rise

Figure 6.18: View from tractor of tall vegetation, person, and small dirt mound

Figure 6.19: Infrared data showing high temperature of person and dirt mound

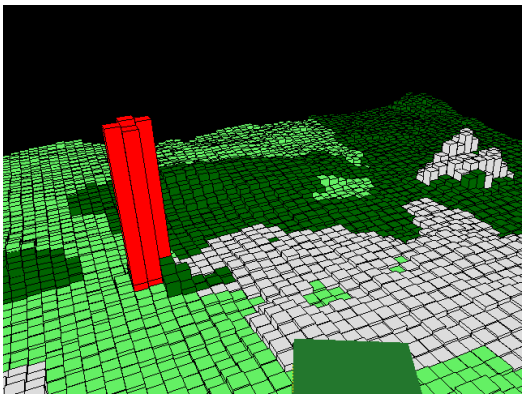


Figure 6.20: Ground height predictions

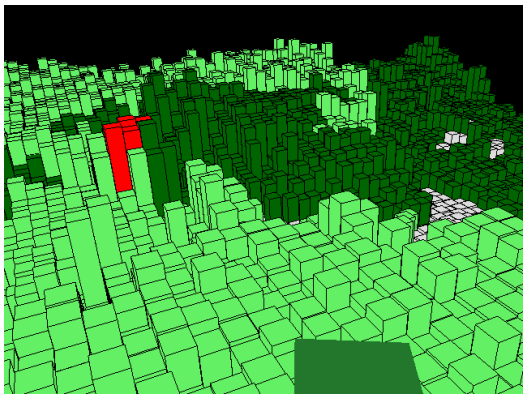


Figure 6.21: Class height predictions

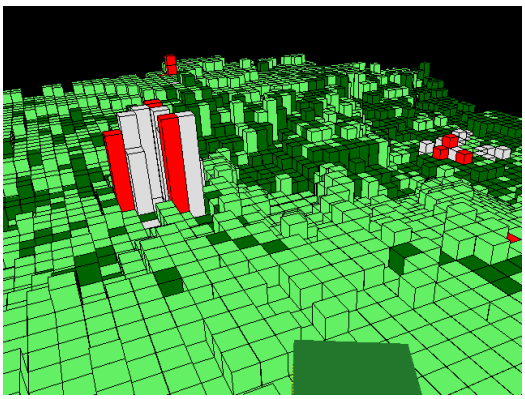


Figure 6.22: Lowest point with independent classifications

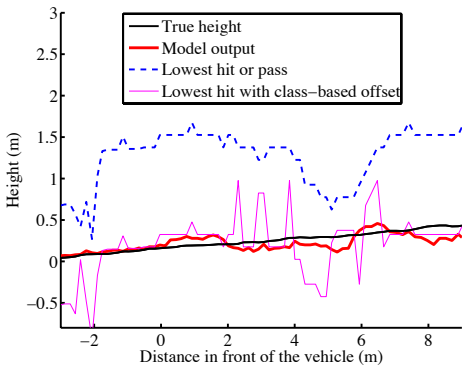


Figure 6.23: Ground height prediction comparison

in the ground heights in Figure 6.20.

Figure 6.22 shows the lowest point and the class estimates when no neighborhood information is used. The lowest point does not penetrate the dense vegetation so it gives poor estimates of the ground height. Both the dirt mound and the person are classified as a mixture of dirt and obstacle columns. Also the vegetation class predictions without neighborhood information are noisy, which causes problems when the class is used to adjust the lowest point as in Figure 6.23.

Figure 6.23 shows a plot of the quality of the ground height estimates from Figure 6.20. After computing estimates of the ground height using our model, we drove through the scene toward the area between the person and the dirt mound, and made measurements of the ground height using our wheel locations. This trajectory is marked as "True height" in Figure 6.23, and offers a comparison for the estimates produced by the model and those using the lowest hit or pass-through in each column, or the lowest hit adjusted by the average trained height of the column ML class. The class adjustment strategy works well when the classifications are correct, but the noise in the classifications creates noise in the ground height estimates. The model ground estimates are fairly smooth and stay within approximately 20cm of the true value.

6.8.5 Vegetation on a Slope

The transition to vegetation result in section 6.8.2 and the person in vegetation result in section 6.8.4 both showed improved ground height predictions. However, the ground was generally flat in both of those examples so a system that simply classified all vegetation as safe could have succeeded without correctly estimating the ground surface (although detecting positive obstacles such as a person is more difficult without the ground plane). This section shows an example where treating vegetation as drivable could lead to dangerous behavior but finding the supporting ground surface enables the vehicle to stay safe.

Figure 6.24 shows the view from the tractor as it is driving along a path on a steep 14-degree side slope with dense vegetation at the bottom of the hill. The vegetation at the bottom covers and hides the supporting ground surface underneath. Figure 6.25 shows a view from in front of the tractor of the range point data, as well as an approximate indication of the true ground height. The path that the tractor is driving on has a high side slope, and the ground becomes even steeper on the lower part of the hill under the vegetation, which could result in a roll-over hazard. The dense vegetation prevents laser measurements of the ground, so a system that uses the lowest point for the ground height would treat the top of the vegetation as the supporting surface, as shown in Figure 6.27. This would make that part of the hill appear to be fairly flat and traversable, when it actually contains a steep side slope and could represent a roll-over hazard.

Figure 6.26 shows the spatial model ground height estimates. This approach correctly infers a ground surface below the vegetation, and the resulting ground surface predicts a high slope in that area that could be used with a vehicle model to check for roll-over conditions.

The model assumptions of smooth ground and similar vegetation height enable the system to infer the ground surface below the vegetation, even though the system was never trained on the height of the vegetation. As in the simulation example in section 6.7, the transition from ground to vegetation at the edge of the path allows the system to infer a

Figure 6.24: View from the tractor on a steep side slope with vegetation at the bottom of the hill on the left

Figure 6.25: Data from the front, showing the path and vegetation on the hill with the approximate true ground height

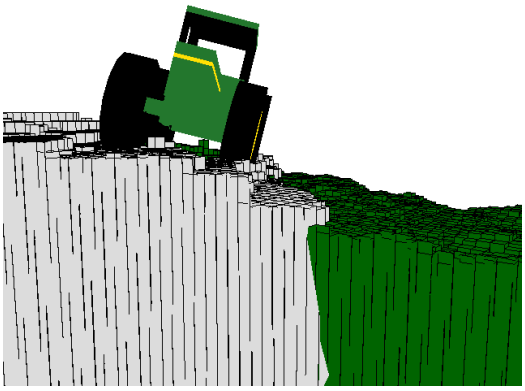


Figure 6.26: Spatial model ground height predictions and classification

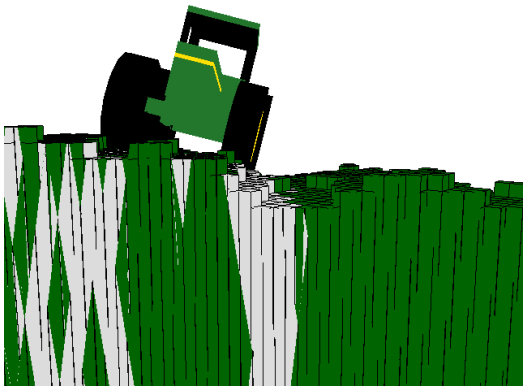


Figure 6.27: Lowest hit or pass-through and independent classification (the diagonal pattern in the classifications is a rendering artifact)

vegetation height which then propagates through the spatial correlations in the model to drop the ground surface below the top of the vegetation that is observable. The system also tries to produce smooth ground estimates between the observed ground height on the path near the tractor and the data from the bottom of the slope (not visible in the figures). These constraints combine to produce an accurate ground estimate in this difficult example.

Figure 6.28: Tractor approaching ledge hazard from the top (vegetation was recently mowed)

Figure 6.29: Ledge with vegetation present during testing

6.8.6 Ledge Step Hazard

The previous sections looked at examples where the model assumptions of smooth ground, class continuity, and similar vegetation height were generally correct. This section explores what happens when model assumptions are broken.

Our main test area did not have any areas with non-smooth ground so a wooden ledge was constructed. Figure 6.28 shows the tractor approaching this ledge from the top. In that picture, the area had been recently mowed, but the data for this experiment was taken when the ledge was more overgrown with vegetation, as shown in Figure 6.29.

The system was trained as in section 6.8.2 on ground and two types of vegetation. Results are given below for the positive obstacle case when the vehicle approaches the ledge from the bottom and the negative obstacle case when the vehicle approaches the ledge from the top.

Ledge from the Bottom

Figure 6.30 shows the view from the tractor as it approaches the ledge from the bottom, where it is directly observable. Figure 6.31 gives the predicted class heights, showing the top of the vegetation. The ledge has an appearance that matches the ground class, so there is a row of columns classified as ground, but the ground prior makes the taller portions of the ledge unlikely to be ground and their appearance does not match any other class, so the ledge is correctly classified as an obstacle.

Figure 6.32 gives the model ground height estimates. The ground estimates beyond the ledge are significantly lower than the true ground height. The model has explained the higher data points beyond the ledge with dense medium height vegetation instead of higher ground and short vegetation.

Figures 6.33 and 6.34 show why the model produces this estimate. The vehicle is positioned in vegetation of the same class as the vegetation beyond the ledge. However, the vegetation that the vehicle is on top of is taller than the vegetation beyond the ledge.

Figure 6.30: Ledge from the bottom

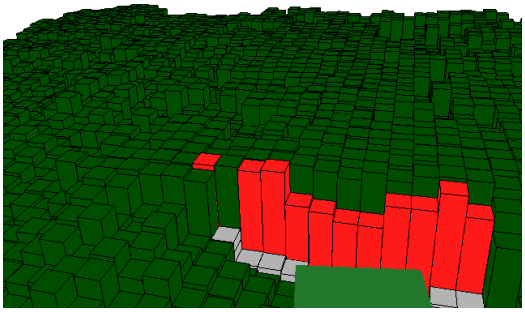


Figure 6.31: Class height

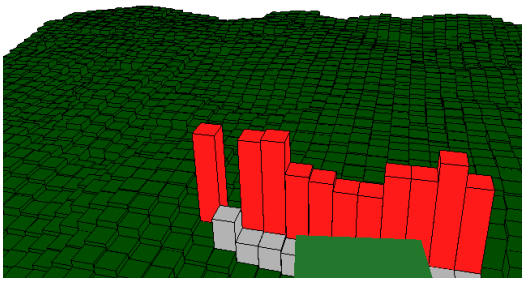


Figure 6.32: Ground height

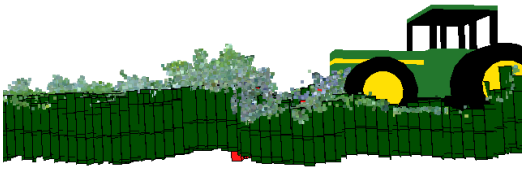


Figure 6.33: Class height (side view)

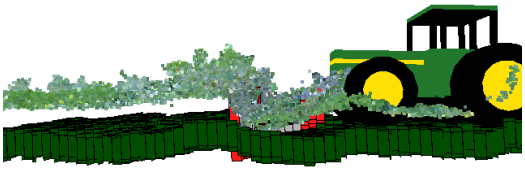


Figure 6.34: Ground height (side view)

Figure 6.33 shows that the system uses the assumption of similar vegetation height to propagate the vegetation height under the vehicle to the area beyond the ledge. The ground rises accordingly but not to the actual ground height.

Although the ground heights beyond the ledge are incorrect, the system correctly detected the ledge as an obstacle because it is inconsistent with the ground smoothness prior.

Ledge from the Top

Figure 6.35 shows the view from the tractor as it approaches the ledge hazard from the top. Detecting negative obstacles such as this is difficult even for a human. Figures 6.36 and 6.37 show the colored data points from above and the side. The large area with no data corresponds to the location of the ledge.

Figure 6.38 shows the system ground height predictions. The ground height is constrained under the vehicle and the system makes predictions for the terrain where it has data. It then smoothly interpolates between the two to produce ground predictions where there is no hit data (the area above the ledge still contains pass-throughs which give an upper bound on the location of the ground surface).

When the vehicle approached the ledge from the bottom, the system received large amounts of data on the ledge that the system was forced to explain. This resulted in the system maintaining the smooth ground and explaining the data with the obstacle class. In this example with the vehicle approaching from the top, there is no data on the ledge for the model to explain, so the model just produces the minimum energy solution that is consistent with the data, which is a gently sloping surface instead of an abrupt ledge.

Figure 6.39 shows the result using the lowest hit or pass-through and the independent column classifications. The classifications have more noise and the vegetation to the left of the tractor's front wheel is not removed, but the ground heights at the ledge are very similar to the model predictions in Figure 6.38. This again shows that the data for this problem does not show the existence of the ledge, which is why negative obstacles like this are so challenging.

One approach that could help the model deal with situations like this is to include a "hole" class in addition to the ground, vegetation, and obstacle classes. A hole class would use an HSMM model that transitioned from the ground state to free-space to free-space, and the output ground height would be the transition from free-space to free-space (the "top" of the hole would be at ground level). This would allow the model to interpret negative obstacles as an area with free-space below the surrounding ground. By including this general class of obstacle into the model, it may be better able to handle these difficult cases.

6.9 Summary and Limitations

This chapter has presented a novel model structure that allows multiple Markov random fields to interact through a hidden semi-Markov model for improved ground height estimation and classification for outdoor navigation. This structure enforces spatial constraints within a column and between neighboring columns, allowing the model to infer vegetation height and use this to find the ground height in dense vegetation. The model provides a natural way of combining different types of sensor data. It can find obstacles without

Figure 6.35: View from the tractor as it approaches the ledge from the top

Figure 6.36: Data from above

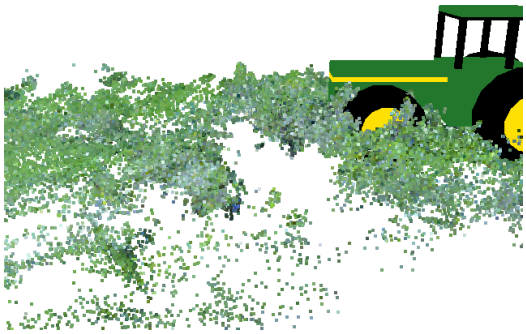


Figure 6.37: Data from the side

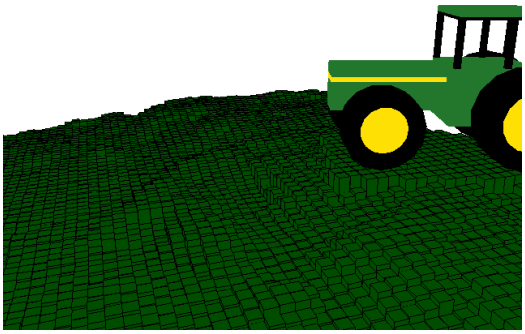


Figure 6.38: Ground height estimates

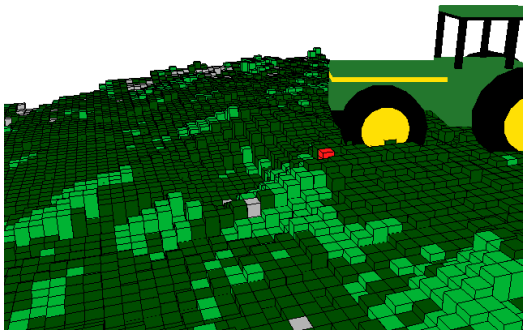


Figure 6.39: Lowest hit or pass-through and independent classification

needing to explicitly model them or collect obstacle appearance training data. Except for the class neighborhood prior, the sensor and interaction model parameters can be easily trained by simply driving through representative areas. The system runs on real data and we showed that including the neighborhood structure significantly improved the ground height estimates over an equivalent model without neighborhood interactions. The fast HSMM calculations and quick convergence of the Gibbs sampling allows the system to be run in real time for moderate vehicle speeds. Chapter 7 gives a comparison between this approach and the online independent learning approach given in chapter 4.

6.9.1 Limitations and Future Work

Although the model can generally run in real time, there are cases when the Gibbs sampling procedure mixes slowly. This often happens when there is very little data (for example, the area behind the vegetation that was classified as ground in Figure 6.20). In this case, all classes are equally probable, but the system will converge to one class in this region and then the class prior will make it difficult for any single column to change. This is a classic problem in sampling methods and can be addressed by the Swendsen-Wang sampling procedure [Swendsen et al., 1992] which extracts groups of like classes and samples a change to all of them at once instead of sampling each individually.

We would like to try adding new general classes such as the "hole" class described in section 6.8.6 or an "overhanging obstacle" class to handle tree branches. It would also be interesting to try to detect when the model is a poor fit for the data to better handle unforeseen situations.

The system is set up essentially as a batch process, even though data is continually coming in and the sampling procedure can continue over time. We would like to make it a true online algorithm like the approach presented in chapter 4. This is discussed in more detail along with other future work in chapter 8.

Finally, we would like to look into other approximate inference schemes that might be less computationally intensive than Gibbs sampling.

Chapter 7

Comparison

This chapter provides a comparison between the two approaches presented in this thesis:

- Independent online learning (Chapter 4)
- Spatial model learning (Chapter 6)

The approaches will be evaluated in various types of vegetation by comparing their ground height predictions at different distances in front of the vehicle with the true ground height found when the vehicle drives over that area.

The test area and training procedures are described below, and then two comparison experiments are presented. The first test compares the results for the transition to dense vegetation example that has been described in earlier chapters individually for each approach. The second test describes a longer experiment with varied vegetation and gives summary results. These results show that both approaches produce better ground height estimates than prior work which is based on the deepest range penetration (lowest hit or pass-through). The results also show that the second approach that includes spatial correlations produces smoother ground estimates with smaller errors than the first approach that treats terrain patches independently.

7.1 Test Environment and Training

These experiments were performed in an undeveloped field with various types of vegetation, as shown in Figure 7.1. This area has two predominant types of vegetation: tall yellow weeds and shorter green grass. Figure 7.2 shows a transition between these types of vegetation as well as some sparse white vegetation on the left.

Data collected during a 13 minute drive through part of this area was used as training data. The independent online learning approach from chapter 4 was simply driven over this area to train it using features from each column of data (see section 4.1). The spatial model approach from chapter 6 requires separate training for each class, so the training data was split between the sections in the tall yellow vegetation and everything else (i.e. miscellaneous vegetation such as the sparse white vegetation in the left of Figure 7.2 was all lumped into the "short green grass" class for simplicity). The sections for the different classes provided labeled voxels for training as well as the spatial correlation parameters for the model (see section 6.5). Both approaches used features based on color, infrared, laser remission, and range points.

Figure 7.1: Test field

7.2 Transition to Dense Vegetation

This section compares the two approaches for a transition from low grass to tall weeds, as shown in Figures 7.2 and 7.3, which has been one of the main motivations for the work in this thesis. This test case has been described in detail in section 4.4.5 for the first approach and section 6.8.2 for the second approach, so only a brief comparison will be presented here.

Figures 7.4 and 7.5 show the ground height predictions produced by each approach for the area in front of the vehicle, and Figures 7.6 and 7.7 give a side view of the predictions and the raw colored range data. Figures 7.8 and 7.9 compare the predictions for each approach with the lowest hit or pass-through and the true ground height found by driving into the tall vegetation.

These results show that both approaches can make predictions of the hidden ground surface below dense vegetation where there are no range measurements. However, because the first approach assumes spatial independence and explicitly trains on the vegetation height, its predictions are noisy and have an offset that could cause a false positive that would prevent the vehicle from entering the tall weeds. The spatial model encodes ground smoothness and can infer the vegetation height so it produces accurate and smooth ground height estimates that would allow the vehicle to enter the tall weeds.

Figure 7.2: Transition to tall weeds

Figure 7.3: Color range data

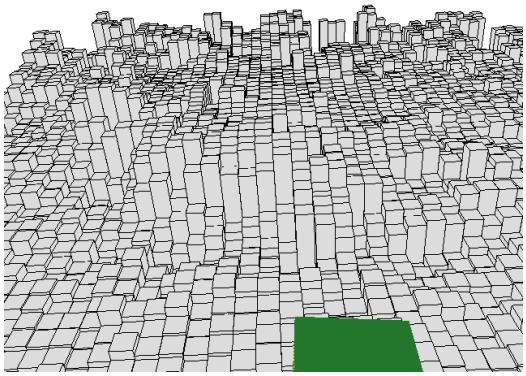


Figure 7.4: Independent ground predictions

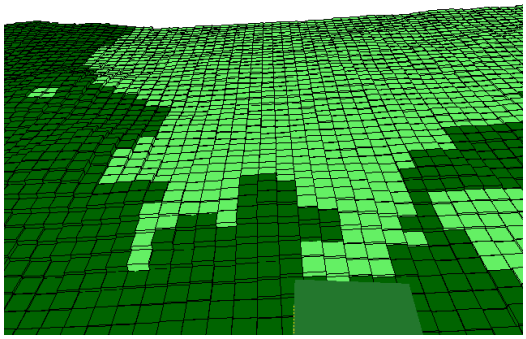


Figure 7.5: Spatial model ground predictions and class

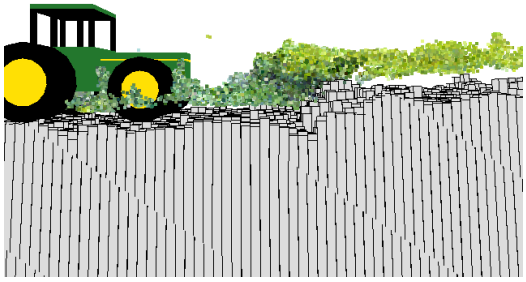


Figure 7.6: Side view showing independent ground predictions

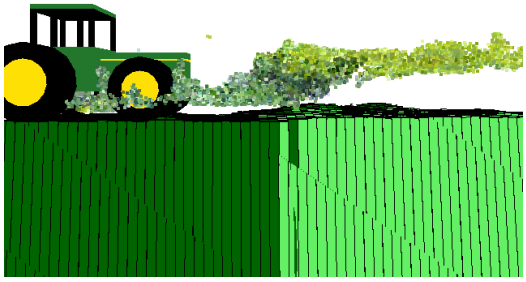


Figure 7.7: Side view showing spatial model ground predictions and class

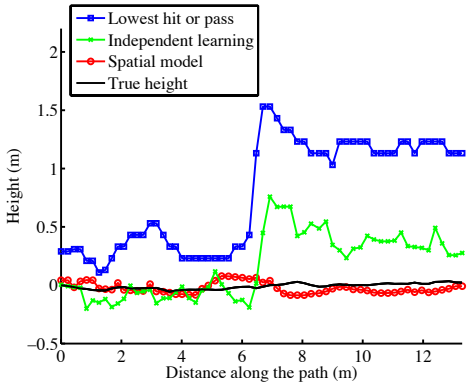


Figure 7.8: Height prediction comparison for left wheel

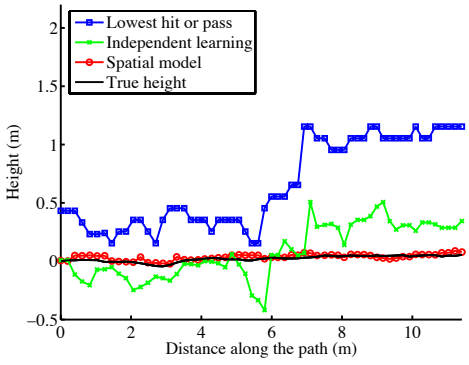


Figure 7.9: Height prediction comparison for right wheel

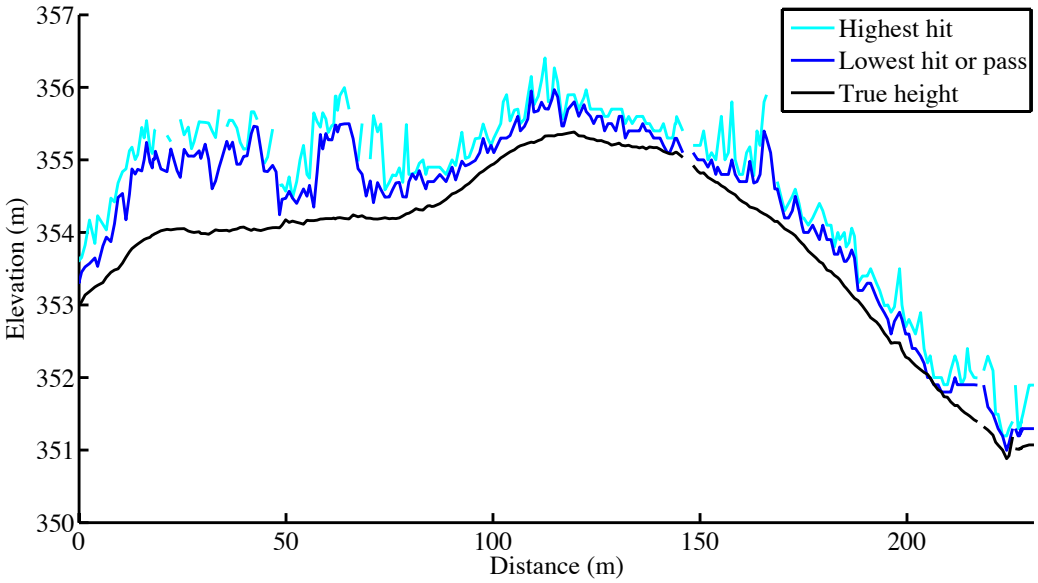


Figure 7.10: Profile of comparison test path (note the two axes have very different scales)

7.3 Long Test Set

Figure 7.10 shows a profile of the test path, along with the lowest and highest hit features to give some indication of the type of vegetation along the path. The path is gently sloping with an elevation change of approximately 6m. The lowest point feature shows that the first 70m of the path contains two sections of tall dense non-penetrable vegetation (the transition example in section 7.2 occurs at 55m). The remainder of the path has low vegetation with various tall sparse vegetation and a few small patches of dense vegetation (e.g. 170m).

The following algorithms were tested on this data set:

Lowest hit or pass Deepest penetration of the range sensor

Lowest w/class adjustment Deepest penetration of the range sensor with an adjustment based on the independent classification of that column and the average trained height of that class (this algorithm uses the same learned sensor models for classification as the spatial model approach but with no neighborhood information - see section 6.8.1)

Independent online Online learning approach presented in chapter 4

Independent online with no adaptation Online learning approach presented in chapter 4 with no online adaptation during testing

Spatial model Spatial model learning approach presented in chapter 6

Flat predictions Always predicts the ground is flat based on the current vehicle location

We present results for different prediction distances in front of the vehicle in summary and detailed form. Figures 7.11 to 7.16 summarize the results on this test set by giving the spread of deviations from the true height for various prediction distances in front of the vehicle. The plots give two sets of error bars. The outer bars give the minimum and maximum deviation from the true height, and the inner bars give the $\pm 2\sigma$ spread. A good

approach should have a zero mean and a small spread of deviations from the true height for all distances in front of the vehicle.

Figures 7.17 to 7.22 give the same results in a more detailed form. The actual deviations from truth are given along the entire path for various prediction distances in front of the vehicle. The following sections describe these results in more detail.

7.3.1 Lowest and Lowest with Class Adjustment

Figure 7.17 shows that one cannot rely on range data penetration in dense vegetation, and this is also reflected in the large positive offset and wide spread in Figure 7.11. Using the lowest range point will cause many false positives and will treat tall weeds as an obstacle.

Figure 7.18 gives the results for subtracting a trained offset from the lowest point that is based on an independent column classification using the learned sensor models from chapter 6. When the classification is correct (for example the tall vegetation from 10-40m for small prediction distances), this technique does fairly well, but without spatial context the classifications are often wrong which results in noisy and incorrect ground estimates because the wrong offset is applied to the lowest point. The summary in Figure 7.12 shows a mean closer to zero than using the lowest point directly, but an even larger spread than simply using the lowest range point.

7.3.2 Independent Online Learning: With and Without Adaptation

The results for the independent online learning approaches summarized in Figures 7.13 and 7.14 have zero mean and a fairly small spread but contain a number of large positive errors. Figures 7.19 and 7.20 show that these large errors generally occur in tall vegetation. Since the independent classifications in Figure 7.18 have large positive errors in many of the same locations (e.g. 25m, 70m, 170m), it is likely that these errors are due to ambiguous feature data that resulted in the wrong vegetation height offset being applied. The independence assumption also results in noisy estimates that could cause many false positives.

For this test set, adapting online during the test does not seem to provide much benefit. This shows that the training data was sufficient for this task so that adding more training data during the test was not necessary, and suggests that for these features in this environment, making independent predictions is likely to have occasional large errors.

7.3.3 Independent Online Learning versus Spatial Model

The results from the second approach that includes spatial correlations and can infer vegetation height are summarized in Figure 7.15. The deviations are zero mean and have a low spread like the independent online learning approach, but the maximum errors are also small. Figure 7.21 shows that the ground height estimates are smooth and would be well suited for vehicle navigation.

Comparing these results with the results for the first approach that makes predictions on an independent basis (Figures 7.19 and 7.20) shows the benefits of including spatial structure to help constrain the problem and handle ambiguous data. The ground predictions from the spatial model are smoother and do not have the tall spikes that can cause false clearance hazards. This is partly due to the class continuity assumption that filters out

isolated misclassifications due to ambiguous data, but even when the classification in the spatial model approach is consistently wrong such as the incorrect classification of the tall weeds at 170m, the ground prior acts to keep the ground predictions smoother and the maximum deviations from the true height significantly smaller than those produced by the independent online approach.

Both approaches in this thesis use learning to differentiate between areas where sensor measurements of the ground are available and areas where the ground is hidden. However, the independent online approach directly applies a learned offset to the lowest sensor measurements based on this differentiation, whereas the spatial model approach relies on spatial structure and model assumptions to infer the offset from the data. If we apply the spatial model approach but turn off all neighborhood connections, then the system can only search for a ground transition using the data in that specific column, which produces estimates similar to the lowest point results shown in Figure 7.17. This system could also classify independent columns based on their sensor data and apply a trained offset based on this independent classification as was done in Figure 7.18, but these results are more noisy than the independent online approach because there is a single offset for each of the discrete classes instead of the continuous space of offsets used in the first approach. The real power of the second approach comes from including the spatial constraints in the model.

7.3.4 Spatial Model and Flat Predictions

As a sanity check, we compare the spatial model with a method that always produces flat predictions based on the vehicle's current height. This method completely ignores the sensor data and always claims the world is safe so it would be worthless to use in a real system, but it provides a good baseline to check how easy this test set is and whether the spatial model is actually doing something intelligent.

The summary results in Figure 7.16 for the flat predictions approach show that the technique works well for short prediction distances but becomes poor as the predictions are made farther in front of the vehicle. This makes sense since the ground in this test set (and in general) is fairly smooth so the vehicle's current height will be a good predictor of the nearby ground height. Figure 7.22 shows that for larger prediction distances when there is a slope, making a flat prediction results in large errors. For example, compare the area around 200m which is on a downward slope. The spatial model prediction errors in Figure 7.21 are near zero whereas the flat predictions errors in Figure 7.22 become larger for larger prediction distances. This shows that the spatial model is smoothing the ground estimates but is also using the data to infer where the ground is even on slopes.

7.3.5 Summary

These results show that both approaches presented in this thesis can produce improved ground height estimates when compared to a method that relies on range data penetrating the vegetation (i.e. the lowest hit or pass-through results). These experiments further show the benefits of including spatial correlations to constrain the problem and better handle ambiguous sensor data.

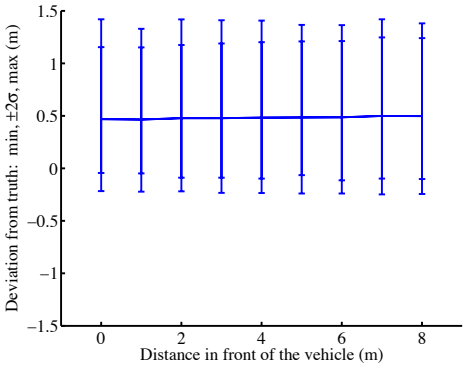


Figure 7.11: Lowest hit or pass

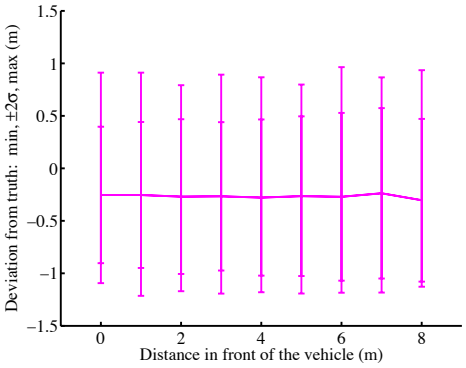


Figure 7.12: Lowest w/class adjustment

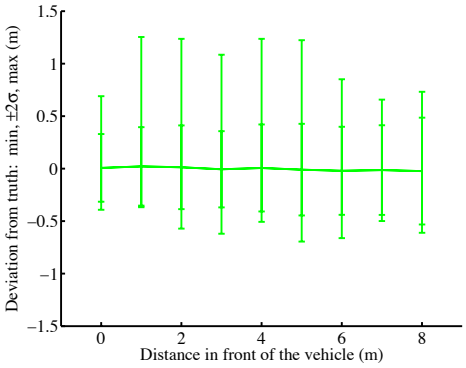


Figure 7.13: Independent online (Ch 4)

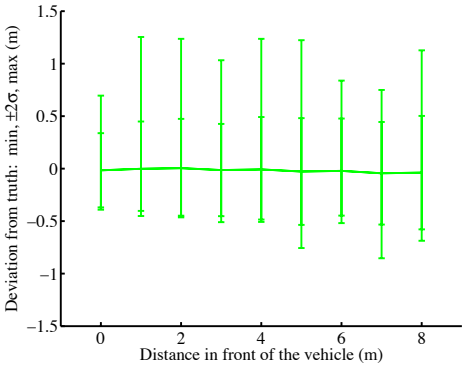


Figure 7.14: Independent online (Ch 4) with no adaptation

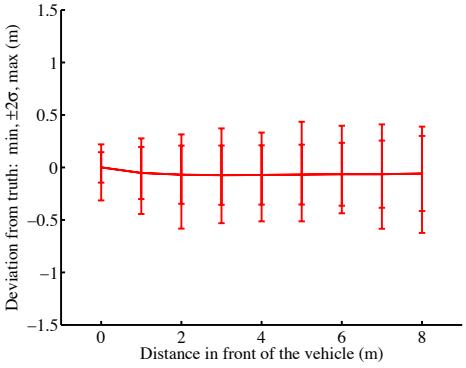


Figure 7.15: Spatial model (Ch 6)

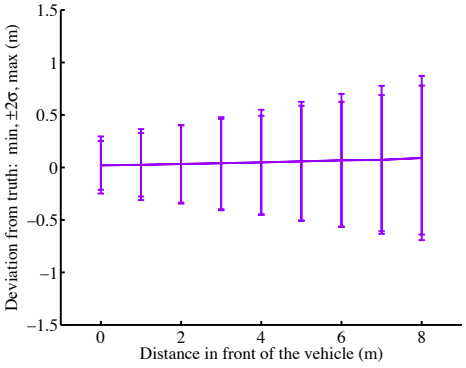


Figure 7.16: Flat predictions

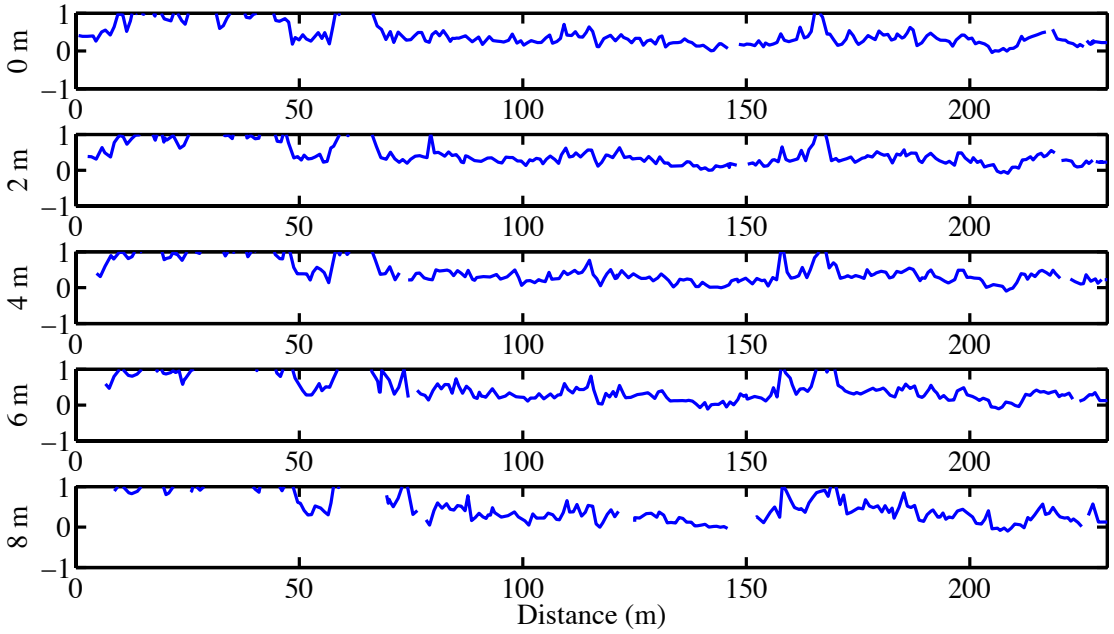


Figure 7.17: Lowest hit or pass
Deviation from truth on test path for different distances in front of the vehicle

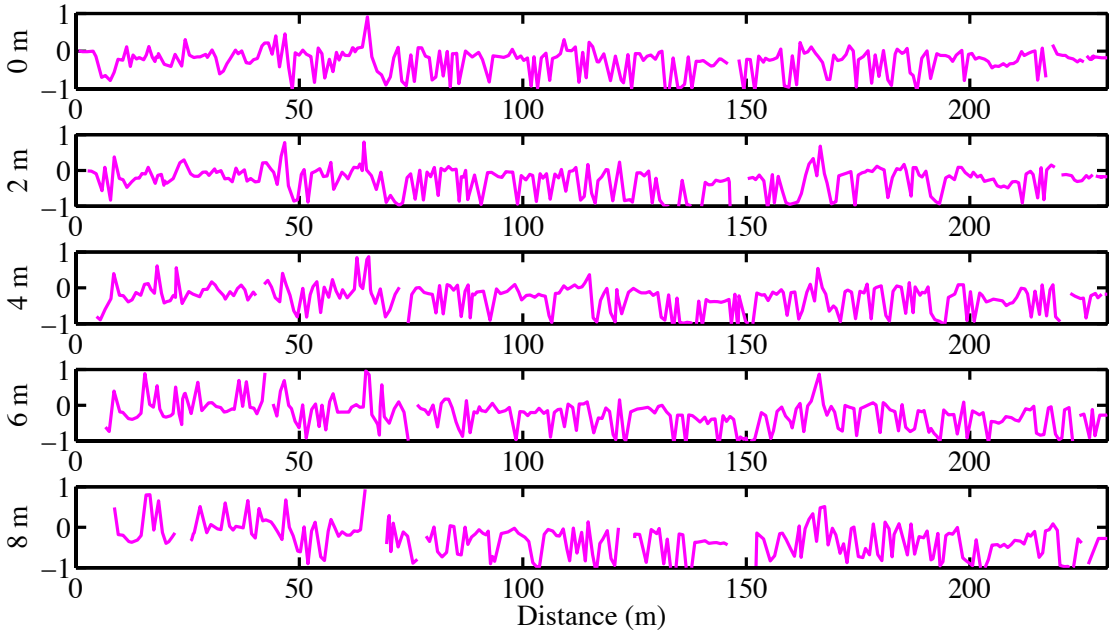


Figure 7.18: Lowest w/class adjustment
Deviation from truth on test path for different distances in front of the vehicle

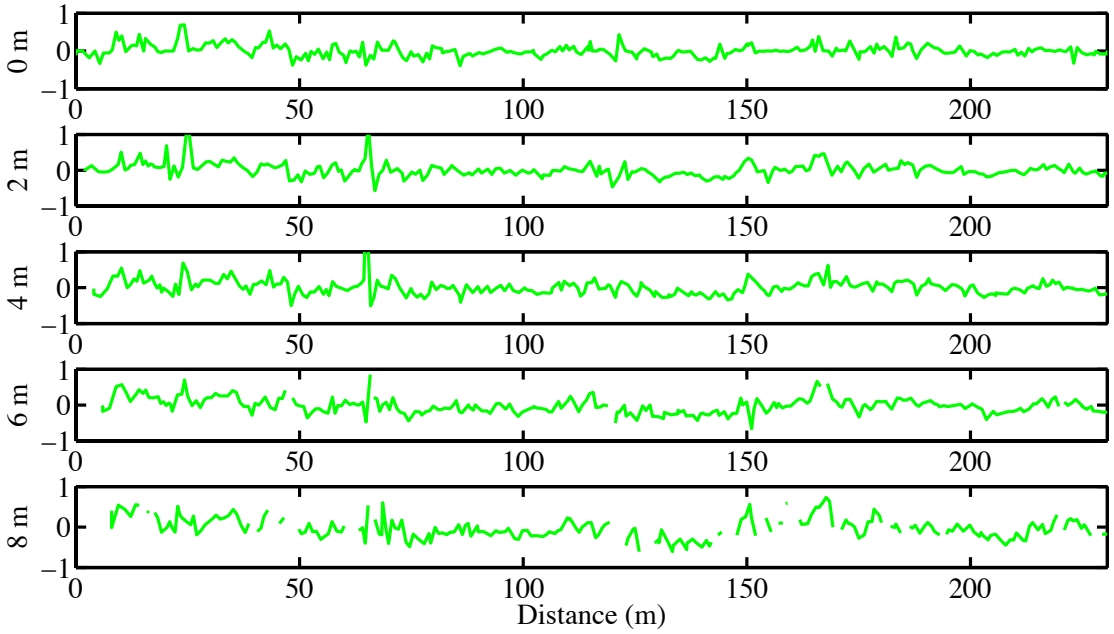


Figure 7.19: Independent online approach (Chapter 4)
Deviation from truth on test path for different distances in front of the vehicle

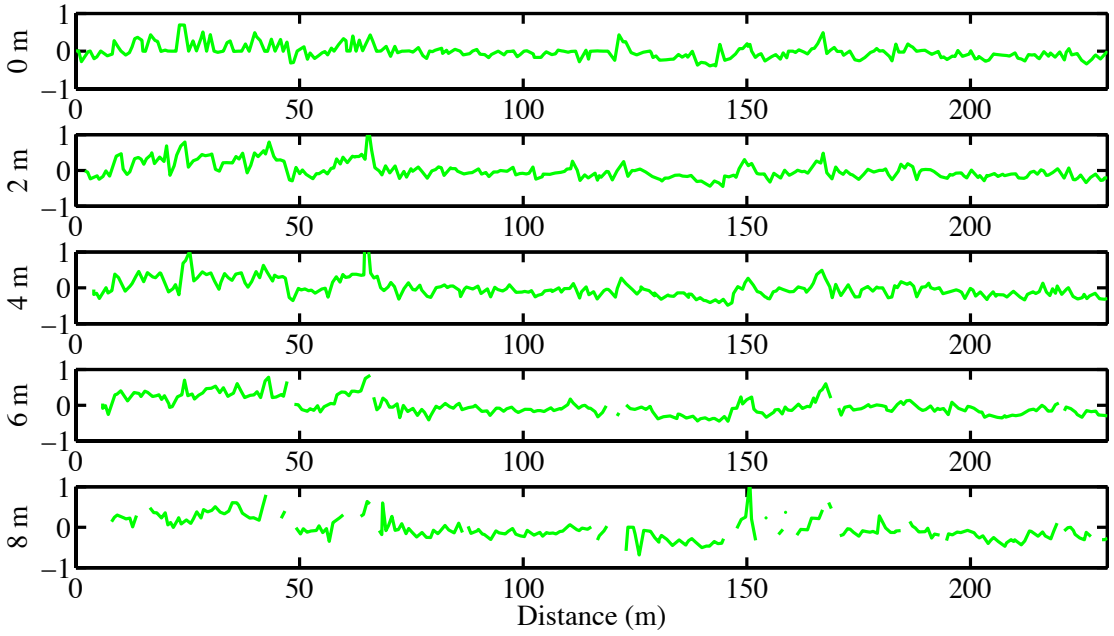


Figure 7.20: Independent online approach - no adaptation (Chapter 4)
Deviation from truth on test path for different distances in front of the vehicle

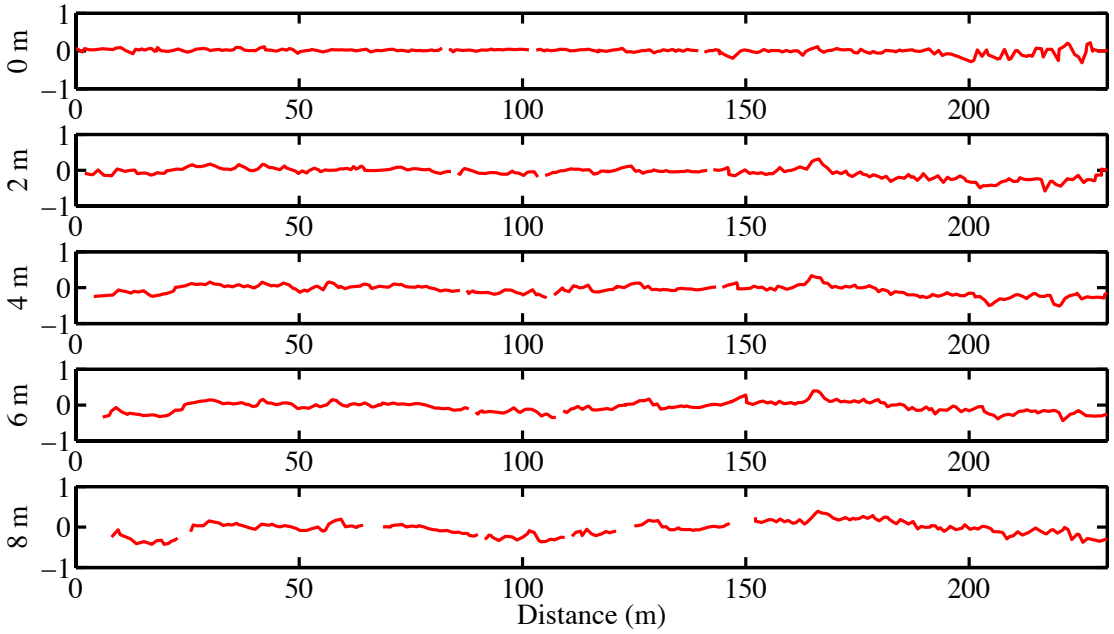


Figure 7.21: Spatial model approach (Chapter 6)
Deviation from truth on test path for different distances in front of the vehicle

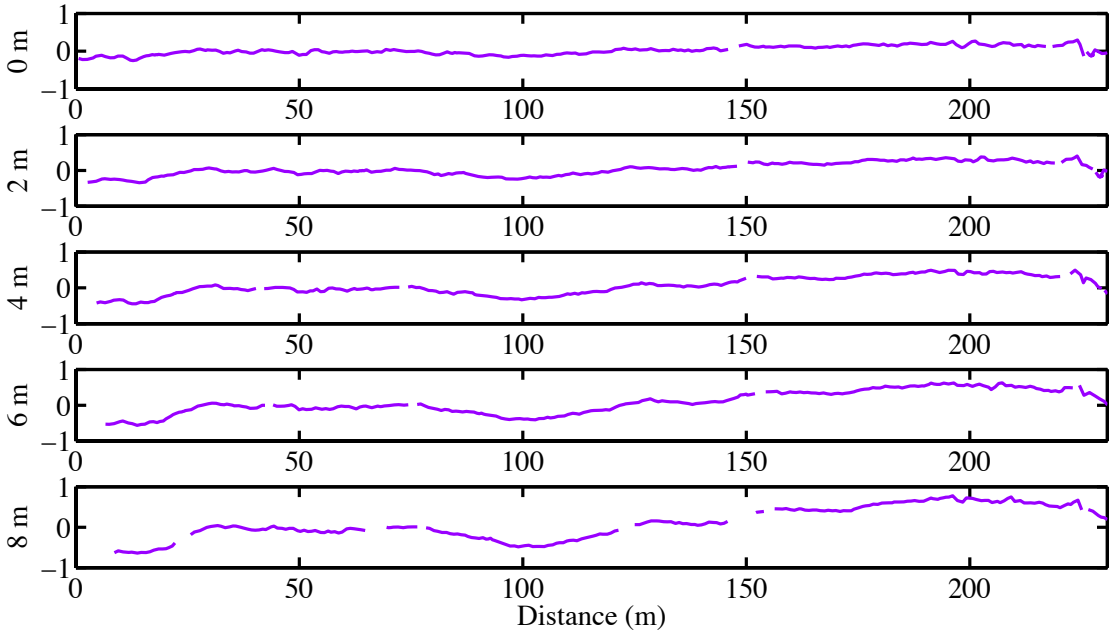


Figure 7.22: Flat predictions
Deviation from truth on test path for different distances in front of the vehicle

Chapter 8

Conclusion

8.1 Summary

This thesis has presented two related approaches for automatically learning a terrain model for autonomous navigation in rough terrain that includes vegetation. The first approach learns the mapping from sensor data to terrain properties online by interacting with the world. The second approach includes spatial correlations to constrain the problem and better handle dense vegetation and ambiguous data. Both approaches have been implemented on an autonomous vehicle platform and can run in real-time for moderate vehicle speeds of 1-2m/s.

8.2 Contributions

- First autonomous vehicle that uses interactions with the world to automatically learn the mapping from range and appearance data to the ground height in vegetation
- New terrain model that exploits spatial correlations to estimate the ground surface when it is hidden below dense vegetation
- Implementation of two terrain model learning approaches on an automated tractor

8.3 Discussion

Machine learning algorithms and probabilistic modeling have been used to create some of the most successful systems in a variety of important domains such as face recognition, handwriting recognition, and speech recognition. These techniques allow assumptions to be included in a clear and natural way, they can be trained using real data instead of relying on tweaking parameters by hand, they can handle uncertainty and produce confidence bounds, and they have well studied algorithms and properties. However, little work has been done to apply these techniques to the rough-terrain navigation problem.

It is hoped that the work in this thesis shows some of the benefits of using learning and probabilistic modeling in this domain, and gives a direction for future work in this field.

8.4 Future Work and Extensions

8.4.1 More General Model Structures

This section compares the model structures used in the two approaches given in this thesis, and proposes an extension using a more general model structure that has the potential to capture the benefits of both. The graphical models are shown in 1D for clarity, but the approaches discussed actually use a 2D model structure.

The first approach presented in chapters 3 and 4 learns the mapping directly from features of the data to ground height for independent terrain patches. The learning mechanism tries to recover the independent conditional model $P(X_i \mid Y_i)$, which is shown graphically in Figure 8.1. Since the states X_i are treated as independent given the data, this model can easily be generalized to use any features of the data by using $P(X_i \mid Y)$, as shown in Figure 8.2. This generalized model is used in chapter 4 to allow more general features, such as the variance from a plane fit over a larger local area and a feature that tries to detect obstacle range shadow by looking at the number of hits in adjacent columns. These features are calculated from the data in the 3x3 area around each X_i, but other features at larger scales that use even more data could be included as well.

Chapters 5 and 6 described a method that includes spatial correlations between state variables instead of assuming independence. This method uses a generative framework where the system must learn the generative observation model $P(Y_i \mid X_i)$ as well as the neighborhood correlations of the Markov random field. The 1D version of this is shown in Figure 8.3. Since generative models condition on the state, the process and observation models become decoupled (see section 5.1). This makes the learning problem much easier since the process (neighborhood) and observation models can be trained individually. However, it also limits the type of features that can be used, since Y_i must be independent of the other measurements given the state. Features that use data from multiple columns as in the first approach are not allowed. This approach also requires us to specify generative models $P(Y_i \mid X_i)$ that describe what type of measurements are typical for each state. This prevents us from using a detector that looks for specific observation patterns to find a certain type of obstacle (as is done in the first approach), and forces us to specify generative models even when they are not natural, such as a model of the typical appearance for an obstacle.

Figure 8.4 shows a model structure that conditions on the data as in the first approach, but includes spatial correlations as in the second approach. This model does not require generative models that can be difficult to specify and allows non-local features of the data. However, it doesn't handle missing data well, and learning in this model can be difficult and often requires large amounts of training data. Recent work in speech recognition [Lafferty et al., 2001] and finding man-made structures in images [Kumar and Hebert, 2003] shows promising results from these conditional random field models, and it would be interesting to look at the terrain modeling and vehicle navigation problem in the context of this model structure, in order to include more general features as well as spatial correlations.

As computational resources continue to increase, it may also make sense to move to a full 3D model with neighborhood connections between every voxel. This type of model is less constrained so it could represent things like horizontal wires or overhanging tree branches, which are not handled well by the methods presented in this thesis.

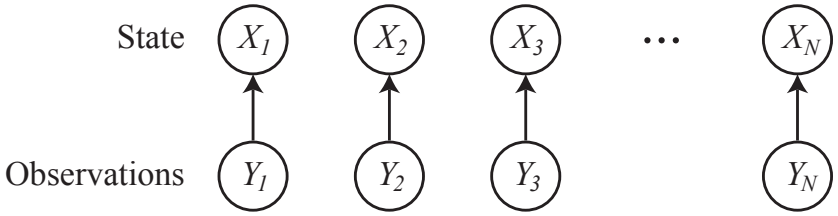


Figure 8.1: Independent conditional model

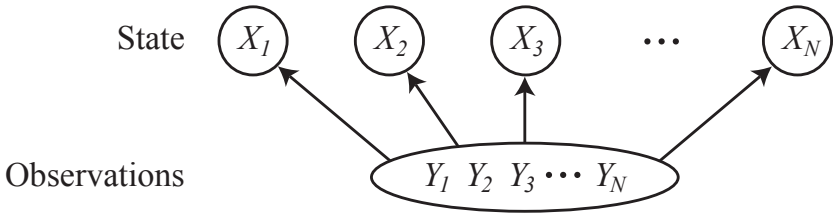


Figure 8.2: Conditional model

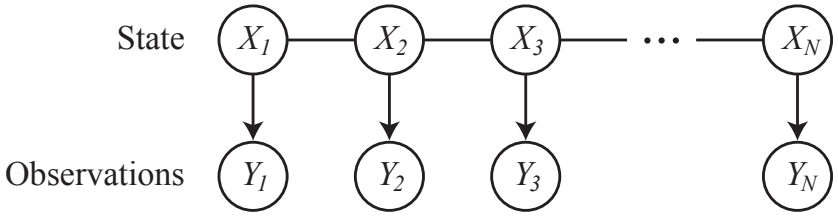


Figure 8.3: Markov random field generative model

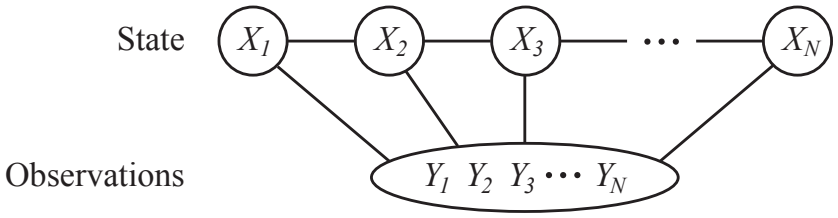


Figure 8.4: Conditional random field conditional model

8.4.2 Online Learning in the Spatial Model

Chapter 3 described the benefits of a system that can adapt online, and chapter 4 gave an example of a system that can adapt automatically to new types of vegetation. The spatial terrain model described in chapter 6 has a number of benefits, but it is currently trained and then runs without adapting. It would be interesting to include online adaptation into the spatial model.

As described in chapter 3, the key requirement for online learning is to be able to make direct measurements of quantities of interest, so that the desired output can be associated with input features to give feedback to the learning mechanism. The spatial model in chapter 6 produces estimates of ground height, class height (e.g. vegetation height) and class. Section 6.5 described how ground height and class height can be measured during training, but it did not give a method for automatically measuring class. The class prior had to specified by hand and the class observation models were trained by driving separately through representative terrain for each class.

The online learning approach in chapter 3 avoids this problem by ignoring the concept of class. However, a key feature of the spatial terrain model is that it exploits the idea that vegetation of a single class has a similar height, so lumping all of the non-obstacle classes such as ground and the various vegetation classes into one single "drivable" class robs the spatial model of its power. To make the spatial model learn online, the system must be able to automatically classify columns of voxels and then feed back this labeled training data to improve the model.

For some domains with easily separable classes, simple heuristics could be used to automatically train the system when the vehicle drives over a terrain patch and finds the true ground height. For example, a domain that only includes smooth ground and tall vegetation could use a simple rule that checks if all the range points are near the ground or if there are points significantly above the ground and then classify the voxels in that column as ground or vegetation accordingly. However, creating and tuning these heuristics in more complex domains can be difficult and time consuming. In fact, it was the failure to create well-performing heuristics for complex environments that was one of the main motivations for investigating the learning approaches presented in this thesis. We lose much of the benefit of a learning approach that can be automatically trained from data if it requires highly accurate hand-crafted heuristics.

An alternative would be to use the inferred class from the model as truth to allow online learning. This would leverage the existing model structure that has been created instead of requiring new heuristics. With the known locations of the wheels acting as a constraint on the ground surface and the dense appearance measurements near the vehicle, the model class estimates near the vehicle should be much more accurate than estimates farther in front of the vehicle. This suggests that it may be possible to use the nearby classifications as "truth" to produce labeled training data for the sensor models that are used to make estimates farther in front of the vehicle.

8.4.3 Including Prior Map Data

Many applications have prior map data available that could be used as an additional piece of information to help interpret the local sensor data. Including prior map data

has been used in many other problems, such as planning in partially known environments [Stentz, 1994], using a "flying eye" robot to collect sensor data beyond the vehicle's sensors [Stentz et al., 2002b] [Kelly, 2004], and using overhead data to enable higher speed navigation than a vehicle's sensors would otherwise permit [Urmson, 2004].

A remaining challenge in using prior map data effectively is how to combine prior map data with local sensor data. In the context of this thesis, prior map data may be thought of as another feature which could be used as another input to the learning algorithm. In this way, through experience with the world, the system could automatically learn how to weight the prior map data features with other local sensor data features.

8.4.4 Learning Other Properties such as Friction or Vegetation Stiffness

The learning approaches in this thesis have been used to find the ground surface in domains where the ground is hidden by vegetation, but other indirectly observable terrain parameters could also be investigated, such as surface friction characteristics or the stiffness of vegetation.

In many rough-terrain applications, the friction parameters of the upcoming terrain have important implications for vehicle control, but the measurements from forward looking sensors give very indirect measurements of these quantities. When the vehicle drives over an area, it can use its known position and its wheel rotations to determine how much it is slipping and directly measure the terrain friction parameters under the vehicle [Iagnemma et al., 2002b].

The general online learning approach given in chapter 3 could be directly applied to this problem. Instead of learning the mapping from features to ground height as in chapter 4, the system would learn the mapping from features to terrain friction parameters. When the vehicle drives over an area, it could measure the true friction parameters and use this feedback to improve its predictions.

The limitations from treating terrain patches as independent that were discussed in section 4.5 may be even more of a problem when estimating friction because the measurements from forward looking sensors are very indirect and potentially ambiguous. However, friction properties are often similar in local regions, so the class continuity assumption described in chapter 6 could help find contiguous slippery areas even when there is missing or ambiguous data.

The ideas in this thesis could also be used to learn the stiffness of different types of vegetation, which can affect vehicle mobility. The vehicle could drive through different types of vegetation and monitor the torque required as a measure of stiffness. As with ground height or friction parameters, this data could be used as feedback to automatically learn the mapping from features to vegetation stiffness.

8.4.5 Unknown Pose

The methods presented in this thesis rely on good (local) position estimation for accumulating data and correlating features from forward looking sensors to truth data when the vehicle drives over that area. These approaches are designed to handle noisy data, but the measurement errors due to positioning are not explicitly included.

One way to include positioning errors would be to treat each range measurement as a cone instead of a ray. The width of the cone would be dependent on the vehicle orientation uncertainty. Instead of keeping track of discrete hits and pass-throughs in each voxel along a range measurement ray, each range measurement would give evidence of density to all the voxels in the cone of where the ray may actually be. The cross-sections of the cone may be modeled as a Gaussian so that the center of the cone would give the most evidence and the evidence would taper off radially. Also, the cone would be narrower at short distances, so close measurements would give stronger evidence than far measurements, instead of treating them equally as we currently do.

An even more ambitious idea would be to incorporate the ideas from the simultaneous localization and mapping (SLAM) problem into this model in order to use forward sensors for localization as well as perception. The terrain model structure could help constrain the SLAM problem because sensor measurements that are better aligned would be more likely under the model.

8.4.6 Other Sensors

This work has focused on using observable data such as ladar range measurements to infer hidden quantities such as the ground height beneath dense vegetation. However, other sensors such as radar are available that can penetrate this type of vegetation and produce more direct measurements of hidden solid objects. [Ollis and Jochem, 2003] presented first experiments at including radar measurements in addition to ladar measurements within a voxel density representation. Although this thesis has not investigated vegetation-penetrating radar measurements, the techniques described could easily incorporate this information to produce better predictions.

In the first approach, radar measurements would simply be another feature that could be learned from. For the second approach, a generative model would need to be provided that gives the distribution of radar measurements for various substances. These models could be learned automatically by driving through representative terrain in a vehicle equipped with a radar sensor.

8.4.7 Improved Density Representation

The voxel representation used in this work allows the system to keep track of density information by counting the hits and pass-throughs from range measurements. However, this discretization also introduces undesired artifacts. Figure 8.5 shows an overhead view of a voxel on a wall that gets a low density value because part of the voxel contains hits and the rest of the voxel contains pass-throughs. Using this voxel representation results in a solid object like a wall getting a low density characteristic of vegetation. The same problem occurs with the ground surface (change Figure 8.5 to be a side view and change the wall to be the ground). Choosing smaller voxels could help, but smaller voxels require more computation, contain less data to extract features from, and can become uninformative once they become a similar size as the sensor noise.

One option is to maintain the voxel data structure, but improve the measure of density. For example, each voxel could be further subdivided to try to separate cases like Figure 8.5 where the voxel is half hits and half pass-throughs from sparse vegetation where the entire

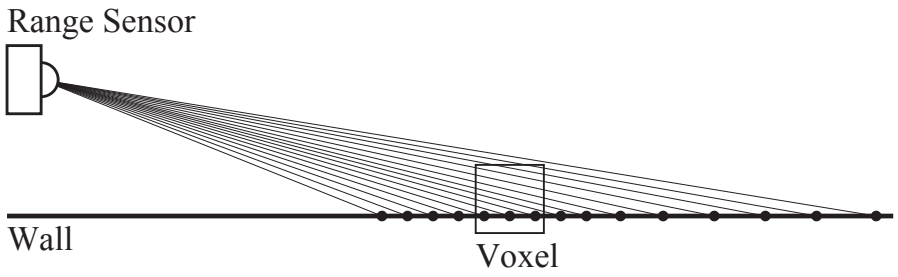


Figure 8.5: Overhead view of range measurements of a wall. The voxel is located on the wall, but has 7 pass-throughs and only 3 hits, giving a low density score.

voxel is a mixture of hits and pass-throughs. However, there are still many questions about how to subdivide and how that would depend on the distance from the sensor and the number of measurements. Another option would be to keep track of the density along different orientations and then use the maximum of these as the density of the voxel. In Figure 8.5, the density parallel to the wall is very low but the density perpendicular to the wall is very high. Sparse vegetation should have low density values along all orientations.

Alternatively, perhaps there is a more general continuous representation for density where each range measurement would produce a line (or cone as in section 8.4.5) of density measurements, or inference could be performed using the measurements directly. Because of position uncertainty and various range measurement errors, a pass-through is only evidence of penetrability, not a hard constraint.

8.4.8 Confidence Estimates for Vehicle Control

Both approaches presented in this thesis can produce prediction intervals on the ground surface estimates, and the second approach that explicitly includes class also produces a confidence measure on the class output. These could be used in a variety of ways for better vehicle control. Vehicle speed could be proportional to the confidence estimates, or the planning system could give preference to high confidence areas, which tend to be roads or other areas with unambiguous measurements.

The ground height confidence estimates could also be used in the calculations of expected vehicle roll and pitch. Instead of just using the mean ground height to find roll, the prediction intervals could be used to find the 95% roll predictions. This would make the vehicle more conservative in areas where its ground estimates are less confident. If the ground estimates are very poorly known, the vehicle will be forced to slow down or stop. This will allow it to accumulate more data which may result in higher confidence predictions that will allow it to proceed. If the system is able to learn online, it may be able to slowly enter a poorly understood area with low confidence predictions, and then as it continues to gain data in the area, its predictions will have higher confidence and the vehicle could start driving faster.

Although this thesis has not investigated the use of confidence estimates for better vehicle control, the ideas described in this section were one of the motivations for looking at terrain modeling techniques that could produce confidence estimates on its output.

Appendix A

System Description

A.1 Sensors

Our project team [Stentz et al., 2002a] [Stentz, 2001] has automated a John Deere 6410 tractor. As shown in figure A.1, this vehicle has a rich set of sensors, including a 2cm differential GPS unit, a 3-axis fiber optic vertical gyro, a Doppler radar ground speed sensor, a steering angle encoder, four custom wheel encoders, a high-resolution (1280x960) stereo pair of digital cameras, an infrared camera, and two SICK laser range-finders (ladar) mounted on custom actively controlled scanning mounts. The first scanning ladar is mounted on the roof to get range data over a large area in front of the vehicle, and the second scanning ladar is mounted on the bumper to get high density measurements of nearby terrain and better penetrate upcoming vegetation. The ladar on the bumper is actively scanned in the direction the tractor is steering.

The cameras and scanned ladars are precisely calibrated and registered with each other in the tractor frame, and the information from the different sensors is tightly synchronized with the vehicle pose to be able to accumulate data into a high quality global terrain map.

A.2 Kalman Filter Positioning Example

This section gives an example of a 2D extended Kalman filter (EKF) used for vehicle position estimation. Our system currently uses an extension of this filter to full 3D pose which results in more complex sensor and vehicle models, but the ideas presented in this section are still relevant to the current filter.

Under the assumptions of white Gaussian noise corrupted measurements and a linear system model, the Kalman filter provides the optimal state estimate. The EKF is an extension of the Kalman filter to handle nonlinear system models by linearizing around the current state estimate.

The EKF utilizes a model of the system to predict the next state of the vehicle. This allows the filter to compare sensor measurements with the expected state of the vehicle and reject sensor measurements that are not consistent based on a likelihood ratio test.

This example uses simple a 2D vehicle model. The model makes two important assumptions. It assumes that the vehicle has the non-holonomic constraint that it can only move forward, not sideways. It also makes a low-dynamics assumption that the forward and

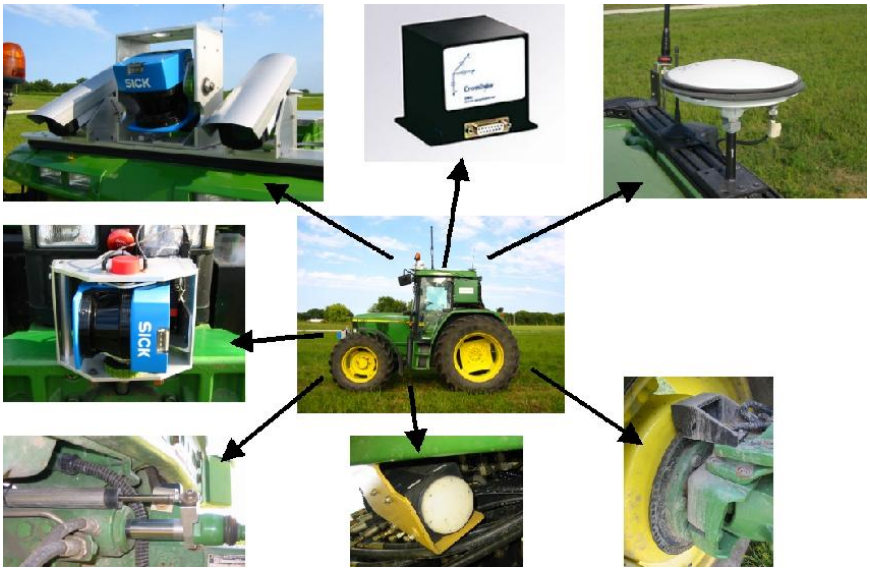


Figure A.1: The sensors on our vehicle - clockwise from top, 3-axis fiber optic vertical gyro, differential GPS, custom wheel encoders, Doppler radar ground speed sensor, steering angle encoder, front bumper scanned ladar, roof scanned ladar and stereo camera pair

angular accelerations of the vehicle are essentially constant over a single time step. Making these assumptions explicit in the model allows the filter to reject measurements that do not follow these assumptions.

The state vector in our model includes global position x, y forward velocity v, global heading θ, and heading rate $\dot{\theta}$. The non-linear system model used in the EKF that obeys the above constraints is given by

$$\frac{d}{dt}\begin{bmatrix} x \\ y \\ v \\ \theta \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\cos(\theta) \\ v\sin(\theta) \\ 0 \\ \dot{\theta} \\ 0 \end{bmatrix} \tag{A.1}$$

where w is a vector of white Gaussian noise representing the uncertainty in the model. The measurements are given as linear combinations of the states

$$\begin{bmatrix} x_{gps} \\ y_{gps} \\ v_{radar} \\ v_{encoder} \\ \dot{\theta}_{encoder} \\ \dot{\theta}_{gyro} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ v \\ \theta \\ \dot{\theta} \end{bmatrix} + r \tag{A.2}$$

where r is a vector of white Gaussian noise representing the uncertainty in the measurements.

Using the noise estimates and models from equation A.1 and A.2, the EKF prediction and correction equations (see section 5.1) are run using whichever measurements are available at a given time step. This is possible because the measurement errors in are assumed to be uncorrelated between sensors. At each time step, the filter outputs an estimate of the state of the system and its associated covariance.

Figure A.2 and Figure A.3 show the behavior of our position estimator for three different types of sensor problems:

- Outliers
- Sensor dropout
- Sensor degradation

Figure A.2 shows an overhead view of a path driven by the tractor. Various sensor problems were simulated during this run. These problems are similar to actual problems we have experienced with our DGPS unit, but simulating them allows a comparison of the filter output to the actual position of the tractor as given by the DGPS. The dashed line shows the DGPS baseline position measurement, while the solid line gives the output of the filter. The x's are the DGPS measurements that were presented to the filter. Every five meters along the path, an ellipse representing the 1-sigma uncertainty in the position measurement is plotted.

Figure A.3 shows the same information in a different format. The center line is the difference between the position estimate and the DGPS baseline as a function of distance along the path. The upper and lower symmetric lines give the 1-sigma uncertainty in the position estimate along the direction of maximum uncertainty.

Near the beginning of the run, two incorrect DGPS measurements were given to the filter. However, since the filter uses its internal uncertainty estimate to test the probability of a measurement being valid, the filter was able to reject these measurements as outliers.

Figure A.3 shows that neither the estimate nor the uncertainty suffered from these false measurements. Other false measurements such as excessive wheel spin causing misleading encoder measurements are similarly filtered out. Prolonged outliers have the same effect as sensor dropout, which is described next. During the first turn, the filter stopped receiving DGPS measurements. The estimate was then based on dead-reckoning alone using the gyro, encoders, and radar. The uncertainty ellipses in Figure A.2 and the uncertainty lines in Figure A.3 show that the uncertainty of the estimate integrates with distance (the uncertainty does not grow linearly because of the curve in the path and the nonlinear filter update). The uncertainty ellipses become wider transverse to the path. This reflects the fact that a small error in the heading becomes large after it is integrated over a long distance. Using these uncertainty estimates, the tractor can find the probability that the vehicle has deviated beyond an application-specific threshold, and stop the tractor if necessary. The remote operator would be alerted to the problem and could correct the situation. However, because DGPS dropout is often caused by occlusion, the vehicle may be able to re-acquire DGPS if it can keep moving to a new location. Figure A.2 and Figure A.3 show this case. The tractor was able to run on dead-reckoning long enough for DGPS to return, at which point the estimate and uncertainty collapsed to their previous levels before DGPS dropped out. After the second turn, the filter is given less precise DGPS measurements. Since our DGPS unit outputs estimates of the variance of its measurements, the filter is able to incorporate the noisy measurements correctly, and effectively low-pass filter them when it combines the DGPS measurements with measurements from the other sensors.

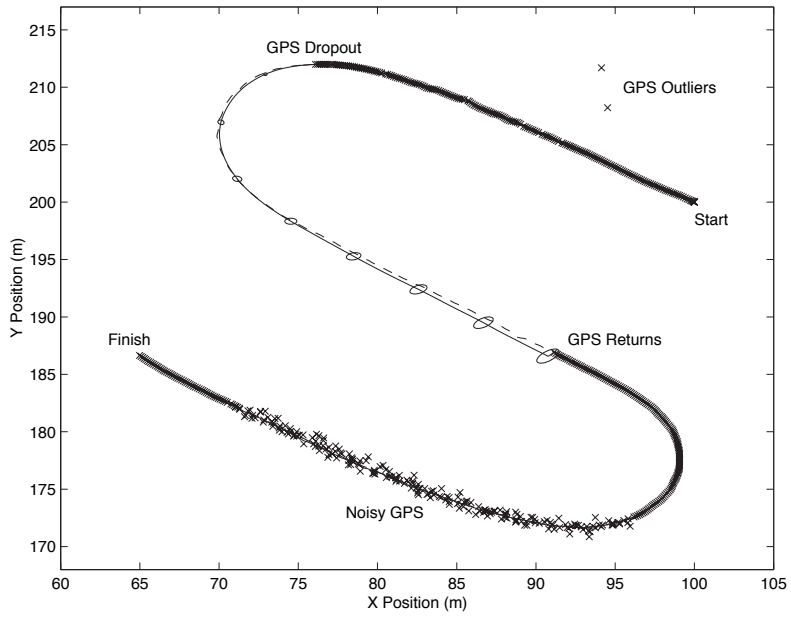


Figure A.2: Position estimation using an extended Kalman filter showing gps outliers, gps dropout, and a degraded gps signal

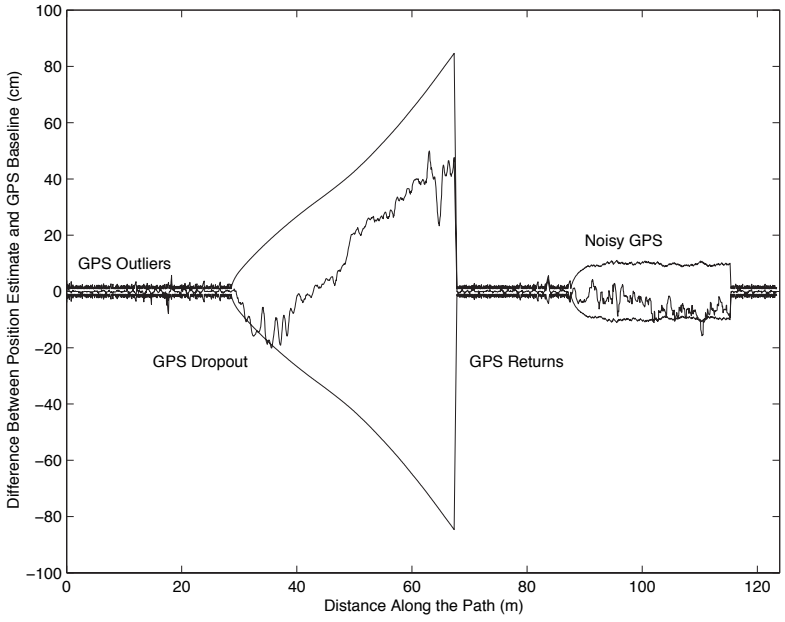


Figure A.3: Position error and uncertainty from figure A.2

Figure A.2 shows that the position estimate remains smooth and close to the baseline despite the noisy measurements. As shown in Figure A.3, the noisy DGPS measurements cause an increase in the position uncertainty, but it remains bounded because of the availability of absolute (noisy) position measurements.

Because the filter is designed to handle sensor dropouts, our system degrades gracefully and will continue to perform well using only DGPS measurements to estimate the entire state vector. Without the high bandwidth of the gyro, the heading estimate will lag somewhat, but more importantly, a DGPS dropout in this scenario would cause the tractor to stop immediately because there would be no other measurements to provide backup information. If the dropout was caused by the geometry of the environment, and the tractor was forced to stop, there would be no chance of DGPS recovery and human intervention would be required.

This section has illustrated two important points. First, using redundant sensors is advantageous because their measurements can be combined to form a better position estimate, and their measurements can be compared to determine if a sensor is malfunctioning. Second, using a filter that produces an uncertainty for its estimate allows the vehicle to make decisions about when it is able to provide reliable position information. This allows the tractor to continue driving autonomously as long as it has determined it is safe to do so. There is often a trade-off between false positives and false negatives. By setting the thresholds conservatively, the filter is able to err on the side of false positives, thereby gaining safety at the expense of a few more interventions by the human operator to check whether the tractor is off course or in a dangerous situation.

Figure A.2 and Figure A.3 have shown that despite the use of encoders, radar, and a gyro, the uncertainty in the position estimate grows when DGPS measurements are not available. This is because DGPS is the only absolute measurement that does not suffer from drift. Also, the majority of the uncertainty is due to errors in the vehicle heading that get integrated over time. We have improved on these results somewhat by extending this filter to a full 3D model and incorporating the other sensors on our vehicle: front-wheel encoders, steering angle potentiometer, and a roll-pitch sensor (this eliminated many of the spikes in Figure A.3 that were caused when the DGPS antenna swung more than 10cm to the side as the vehicle rolled while driving through a rut). However, because none of these measurements give an absolute heading measurement, they still don't bound the uncertainty to allow extended runs without DGPS.

Several options exist to bound the heading error and allow extended runs without DGPS. Magnetic compasses are available that give a direct heading measurement, but these sensors can have large biases when they are near metallic objects. Another option for the structured environments common in agricultural domains is to sense the crop rows and then maintain one's position relative to the crop. For example, we have performed an experiment in an apple orchard where we sensed the tree rows and used these for guidance without using any GPS. For more general environments, techniques from the simultaneous localization and mapping (SLAM) literature could be used [Majumder et al., 2003].

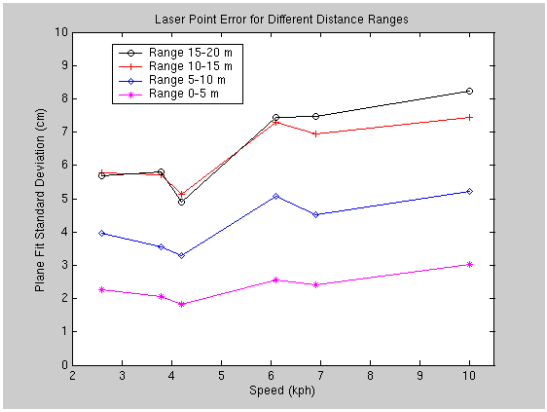


Figure A.4: Plot of height variance as a function of speed for different ranges

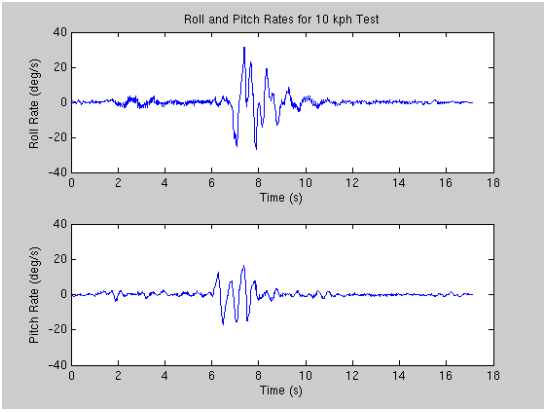


Figure A.5: Roll and pitch rates for the 10 km/hour test run over a log

A.3 Laser Accuracy Bump Tests

The approach described in this work uses points from multiple lasers that are actively scanned while the vehicle moves over rough-terrain. The true ground surface is then found when the tractor drives over that area a number of seconds later. To make this process work, it is important that the scanned ladars are precisely calibrated and registered with each other in the tractor frame, the timing of all the various pieces of sensor data is carefully synchronized, and the vehicle has a precise global pose estimate. We have extended the simple 2D EKF described in section A.2 to a 13 state 3D extended Kalman filter [Maybeck, 1994] with bias compensation and outlier rejection that integrates the vehicle sensors described in section A.1 into an accurate estimate of the pose of the vehicle at 75Hz. This pose information is used to tightly register the data from the ladars into high quality terrain maps. For some robot applications, especially indoors where many clear features such as corners exist, it is possible to perform scan-matching to better register multiple scans together into a more accurate map. However, in outdoor rough-terrain applications that include vegetation, this can be very difficult. We avoid this problem by having good pose estimation that can be used to directly build a map of the environment.

A sequence of tests was performed to determine the accuracy of the maps that are produced. We drove the tractor at different speeds over a 12cm high log on the flat asphalt driveway of the NREC. We then fit a plane to the points collected during the 10 seconds starting just before the tractor hit the log that were also within a 4m wide band of the tractor. Figure A.4 shows the standard deviation of the points from the plane fit. The four lines correspond to four different ranges in front of the tractor. The lowest line only uses points that were very close to the tractor, which have the smallest standard deviation from the plane. The other lines have larger standard deviation because the small errors in tractor roll and pitch get magnified for longer ranges. The plot also shows that the points tend to be less accurate when the vehicle is moving faster. However, even at relatively fast speeds, the mapping result is quite accurate, especially for points near the tractor. Figure A.5 shows the roll and pitch rates for the fastest of these tests (10 km/hour). The pitch rate shows

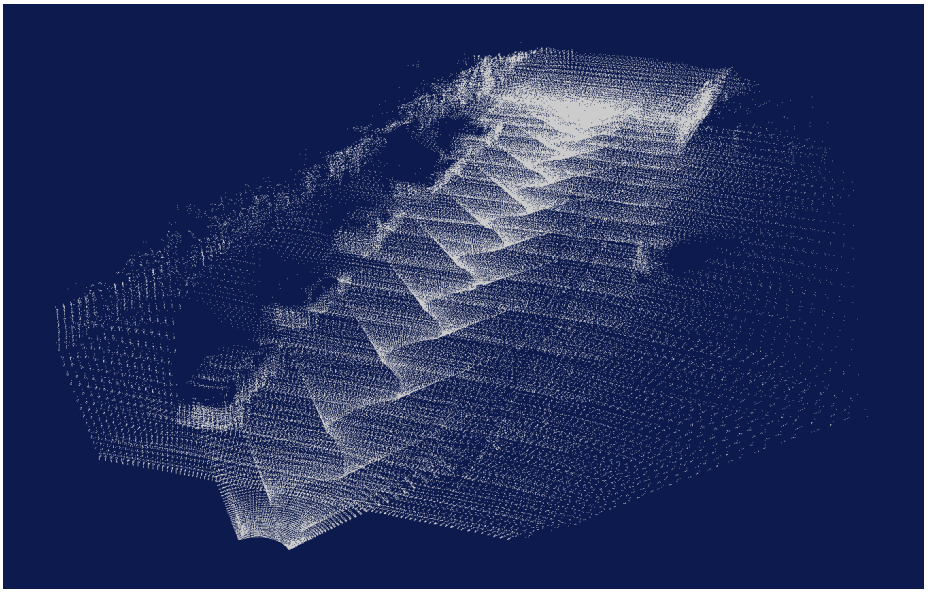


Figure A.6: Map result for the 10 km/hour test run over a log

the front wheel hitting the log, the back wheel hitting the log, and bouncing afterwards. The front wheels are on a rocker suspension, so the roll rate doesn't change until the rear wheels hit the log, but then the tractor bounces back and forth for a couple seconds, and reaches roll rates over 30 deg/sec. Figure A.6 shows the laser points for the 10 km/hour test. The figure shows a number of cars (and their shadows) to show what the terrain maps look like, but these points were not used for the plane fit because they were outside of the 4m wide band. The map result also shows the coverage pattern of the two actively scanned lasers. The serpentine pattern is produced by the front laser that is scanned left and right. The point density is higher at slower speeds, but even at 10 km/hour, the points are spaced dense enough to give good information.

These tests show that even under high speed and relatively violent motion, the points from the actively scanned ladars are accurate and provide useful information in the world frame. This was accomplished through high quality sensors, careful calibration and registration, and sensor fusion using an extended Kalman filter. Our project team has also registered the camera images from the high-resolution stereo pair and the infrared camera with the laser points, which allows us to project the laser points back into the images and attain color and infrared information for the range points as well.

A.4 Low Level Steering Control

Figure A.7 shows the control law for the tractor steering, and Figure A.8 gives its resulting dynamic response. The controller is a proportional controller with a small dead band and minimum/maximum control outputs. The response graph shows that there is approximately a 0.2 second delay before any motion occurs and then it responds as a first order system. The max curvature allowed is 0.188 so for the hard left/right turns, the wheels never reach

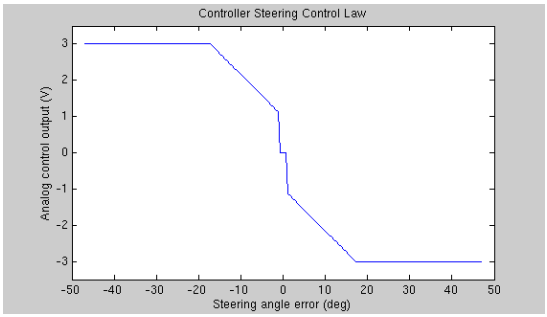


Figure A.7: Low-level steering control law

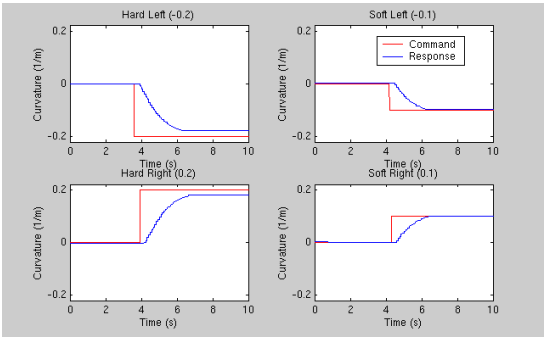


Figure A.8: Steering response to step input

the commanded value. The response curves were taken while moving at 3.1 m/s, but the response is similar at other speeds. The delay and dynamic response shown in Figure A.8 can be incorporated into the model predictive control system described in sections 1.2 and 3.2 to achieve dynamically achievable candidate arcs and therefore stable control.

References

[Glo, 2005] (2005). Demo III experimental unmanned vehicle (XUV) http://www.globalsecurity.org/military/systems/ground/xuv.htm.

[Albus, 2000] Albus, J. (2000). 4-d/rcs reference model architecture for unmanned ground vehicles. In *IEEE Int. Conf. on Robotics and Automation (ICRA 00)*, volume 4, pages 3260–3265.

[Amar et al., 1993] Amar, F., Bidaud, P., and Ben Ouezdou, F. (1993). On modeling and motion planning of planetary vehicles. In *Proceedings of the 1993 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS 93)*, volume 2, pages 1381–1386.

[Amar and Bidaud, 1995a] Amar, F. B. and Bidaud, P. (1995a). Dynamic analysis of off-road vehicles. In *Fourth International Symposium on Experimental Robotics (ISER 95)*.

[Amar and Bidaud, 1995b] Amar, F. B. and Bidaud, P. (1995b). Motion simulation of wheeled off-road robots. In *Proceedings of the Ninth World Congress on the Theory of Machines and Mechanisms*.

[Atkeson et al., 1997] Atkeson, C., Moore, A., and Schaal, S. (1997). Locally weighted learning. *AI Review*, 11:11–73.

[Batavia and Singh, 2001] Batavia, P. and Singh, S. (2001). Obstacle detection using adaptive color segmentation and color stereo homography. In *Proceedings of the IEEE International Conference on Robotics and Automation*.

[Bellutta et al., 2000] Bellutta, P., Manduchi, R., Matthies, L., Owens, K., and Rankin, A. (2000). Terrain perception for Demo III. In *IEEE Intelligent Vehicles Symposium*, pages 326 – 331.

[Besag, 1986] Besag, J. (1986). On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society*, 48(3):259–302.

[Billingsley and Schoenfisch, 1995] Billingsley, J. and Schoenfisch, M. (1995). Vision guidance of agricultural vehicles. *Autonomous Robots*, 2.

[Bouman and Sauer, 1993] Bouman, C. and Sauer, K. (1993). A generalized gaussian image model for edge-preserving map estimation. *IEEE Transactions on Image Processing*, 2(3):296–310.

[Brummit et al., 1992] Brummit, B., Coulter, R. C., Kelly, A., and Stentz, A. (1992). A system for autonomous cross country navigation. In *IFAC Symposium on Intelligent Components and Instruments for Control Applications*.

[Castano and Matthies, 2003] Castano, A. and Matthies, L. (2003). Foliage discrimination using a rotating ladar. In *Proceedings of IEEE Int. Conf. on Robotics and Automation (ICRA 03)*, volume 1, pages 1–6.

[Chanclou and Luciani, 1996a] Chanclou, B. and Luciani, A. (1996a). Global and local path planning in natural environment by physical modeling. In *Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS 96)*, volume 3, pages 1118–1125.

[Chanclou and Luciani, 1996b] Chanclou, B. and Luciani, A. (1996b). Physical modeling and dynamic simulation of off-road vehicles and natural environments. In *Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS 96)*, volume 2, pages 505–512.

[Cherif, 1999a] Cherif, M. (1999a). Kinodynamic motion planning for all-terrain wheeled vehicles. In *Proceedings of the IEEE Int. Conf. on Robotics and Automation*, volume 1, pages 317–322.

[Cherif, 1999b] Cherif, M. (1999b). Motion planning for all-terrain vehicles: a physical modeling approach for coping with dynamic and contact interaction constraints. *IEEE Transactions on Robotics and Automation*, 15(2):202–218.

[Cherif and Laugier, 1995] Cherif, M. and Laugier, C. (1995). Motion planning of autonomous off-road vehicles under physical interaction constraints. In *Proceedings of the 1995 IEEE Int. Conf. on Robotics and Automation*, volume 2, pages 1687 – 1693.

[Coombs et al., 2000] Coombs, D., Murphy, K., Lacaze, A., and Legowik, S. (2000). Driving autonomously off-road up to 35 km/h. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 186–191.

[Daily et al., 1988] Daily, M., Harris, J., Keirsey, D., Olin, D., Payton, D., Reiser, K., Rosenblatt, J., Tseng, D., and Wong, V. (1988). Autonomous cross-country navigation with the ALV. In *IEEE Int. Conf. on Robotics and Automation (ICRA 88)*, volume 2, pages 718–726.

[Davis et al., 1995] Davis, I., Kelly, A., Stentz, A., and Matthies, L. (1995). Terrain typing for real robots. In *Intelligent Vehicles Symposium*, pages 400–405.

[Dima et al., 2004a] Dima, C., Hebert, M., and Stentz, A. T. (2004a). Enabling learning from large datasets: Applying active learning to mobile robotics. In *International Conference on Robotics and Automation*. IEEE.

[Dima et al., 2004b] Dima, C., Vandapel, N., and Hebert, M. (2004b). Classifier fusion for outdoor obstacle detection. In *International Conference on Robotics and Automation*. IEEE.

[Farritor et al., 1998] Farritor, S., Hacot, H., and Dubowsky, S. (1998). Physics-based planning for planetary exploration. In *Proceedings of the IEEE Int. Conf. on Robotics and Automation*, volume 1, pages 278–283.

[Gauss, 1809] Gauss, C. F. (1809). *Theoria motus corporum coelestium in sectionibus conicis solem ambientium (Theory of the Motion of the Heavenly Bodies Revolving round the Sun in Conic Sections)*. Dover, New York. Reprinted 1963.

[Geman and Geman, 1984] Geman, S. and Geman, D. (1984). Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[Gerrish et al., 1997] Gerrish, J. B., Fehr, B. W., Van Ee, G. R., and Welch, D. P. (1997). Self-steering tractor guided by computer vision. *Applied Engineering in Agriculture*, 13(5).

[Goldberg et al., 2002] Goldberg, S., Maimone, M., and Marthies, L. (2002). Stereo vision and rover navigation software for planetary exploration. In *Proceedings of the IEEE Aerospace Conference*, pages 2025–2036.

[Gordon et al., 1993] Gordon, N. J., Salmond, D. J., and Smith, A. F. M. (1993). Novel approach to nonlinear/non-gaussian bayesian state estimation. *IEE Proceedings of Radar and Signal Processing*, 140(2):107–113.

[Hait and Simeon, 1996] Hait, A. and Simeon, T. (1996). Motion planning on rough terrain for an articulated vehicle in presence of uncertainties. In *Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS 96)*, volume 3, pages 1126–1133.

[Hait et al., 1999] Hait, A., Simeon, T., and Taix, M. (1999). Robust motion planning for rough terrain navigation. In *Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS 99)*, volume 1, pages 11–16.

[Hebert et al., 2002] Hebert, M., Vandapel, N., Keller, S., and Donamukkala, R. R. (2002). Evaluation and comparison of terrain classification techniques from LADAR data for autonomous navigation. In *23rd Army Science Conference*.

[Hebert, 1997] Hebert, M. H. (1997). *Intelligent Unmanned Ground Vehicles*, chapter SMARTY: Point-Based Range Processing for Autonomous Driving. Kluwer.

[Iagnemma and Dubowsky, 2002] Iagnemma, K. and Dubowsky, S. (2002). Terrain estimation for high speed rough terrain autonomous vehicle navigation. In *Proceedings of the SPIE Conference on Unmanned Ground Vehicle Technology IV*.

[Iagnemma et al., 1999] Iagnemma, K., Genot, F., and Dubowsky, S. (1999). Rapid physics-based rough-terrain rover planning with sensor and control uncertainty. In *Proceedings of the 1999 Int. Conf. on Intelligent Robots and Systems (IROS 1999)*, volume 3, pages 2286–2291. IEEE.

[Iagnemma et al., 2002a] Iagnemma, K., Golda, D., Spenko, M., and Dubowsky, S. (2002a). Experimental study of high-speed rough-terrain mobile robot models for reactive behaviors. In *Proceedings of the Eighth International Symposium on Experimental Robotics (ISER 02)*.

[Iagnemma et al., 2001] Iagnemma, K., Shibly, H., and Dubowsky, S. (2001). Planning and control algorithms to enhance rover rough terrain mobility. In *Proceedings of I-SAIRAS: 6th International Syposium on Artificial Intelligence, Robotics, and Automation in Space*.

[Iagnemma et al., 2002b] Iagnemma, K., Shibly, H., and Dubowsky, S. (2002b). On-line terrain parameter estimation for planetary rovers. In *IEEE Int. Conf. on Robotics and Automation (ICRA 02)*.

[Kalman, 1960] Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME - Journal of Basic Engineering*, 82(Series D):35–45.

[Kashyap, 1981] Kashyap, R. L. (1981). Analysis and synthesis of image patterns by spatial interaction models. In Kanal, L. N. and Rosenfeld, A., editors, *Progress in Pattern Recognition*. North-Holland.

[Kelly, 2004] Kelly, A. (2004). Toward reliable off-road autonomous vehicle operating in challenging environments. In *International Symposium on Experimental Robotics*, Singapore.

[Kelly and Stentz, 1998a] Kelly, A. and Stentz, A. (1998a). Rough terrain autonomous mobility - part 1: A theoretical analysis of requirements. *Autonomous Robots*, 5(2):129–161.

[Kelly and Stentz, 1998b] Kelly, A. and Stentz, A. (1998b). Rough terrain autonomous mobility - part 2: An active vision, predictive control approach. *Autonomous Robots*, 5(2):163–198.

[Krishnamachari and Chellappa, 1997] Krishnamachari, S. and Chellappa, R. (1997). Multiresolution gauss-markov random field models for texture segmentation. *IEEE Transactions on Image Processing*, 6(2):251–267.

[Krogh et al., 1994] Krogh, A., Brown, M., Mian, I. S., Sjölander, K., and Haussler, D. (1994). Hidden markov models in computational biology: Applications to protein modeling. *Journal of Molecular Biology*, 235(5):1501–1531.

[Kumar and Hebert, 2003] Kumar, S. and Hebert, M. (2003). Discriminative random fields: A discriminative framework for contextual interaction in classification. In *IEEE International Conference on Computer Vision (ICCV)*.

[Lacaze et al., 1998] Lacaze, A., Moscovitz, Y., DeClaris, N., and Murphy, K. (1998). Path planning for autonomous vehicles driving over rough terrain. In *Proceedings of the ISIC/CIRA/ISAS*.

[Lacaze et al., 2002] Lacaze, A., Murphy, K., and DelGiorno, M. (2002). Autonomous mobility for the Demo III experimental unmanned vehicles. In *Assoc. for Unmanned Vehicle Systems Int. Conf. on Unmanned Vehicles (AUVSI 02)*.

[Lacroix et al., 1994] Lacroix, S., Chatila, R., Fleury, S., Herrb, M., and Simeon, T. (1994). Autonomous navigation in outdoor environment: adaptive approach and experiment. In *Proceedings of the IEEE Int. Conf. on Robotics and Automation (ICRA 94)*, volume 1, pages 426–432.

[Lafferty et al., 2001] Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning (ICML)*.

[Lakshmanan and Derin, 1993] Lakshmanan, S. and Derin, H. (1993). Gaussian markov random fields at multiple resolutions. In Chellappa, R. and Jain, A., editors, *Markov Random Fields: Theory and Application*, pages 131–157. Academic Press.

[Langer et al., 1994] Langer, D., Rosenblatt, J., and Hebert, M. (1994). A behavior-based system for off-road navigation. *IEEE Trans. Robotics and Automation*, 10(6):776–783.

[Le et al., 1997] Le, A. T., Rye, D., and Durrant-Whyte, H. (1997). Estimation of track-soil interactions for autonomous tracked vehicles. In *IEEE Int. Conf. on Robotics and Automation (ICRA 97)*, volume 2, pages 1388–1393.

[Li, 2001] Li, S. Z. (2001). *Markov Random Field Modeling in Image Analysis*. Computer Science Workbench. Springer-Verlag, 2nd edition.

[Ljung, 1987] Ljung, L. (1987). *System Identification - Theory for the User*. Prentice-Hall.

[Macedo et al., 2000] Macedo, J., Manduchi, R., and Matthies, L. (2000). Ladar-based discrimination of grass from obstacles for autonomous navigation. In *Int. Symposium on Experimental Robotics (ISER 00)*.

[Majumder et al., 2003] Majumder, S., Durrant-Whyte, H., Thrun, S., and de Battista, M. (2003). An approximate bayesian method for simultaneous localisation and mapping. *IEEE Transactions on Robotics and Automation (submitted)*.

[Manduchi et al., 2005] Manduchi, R., Castano, A., Talukder, A., and Matthies, L. (2005). Obstacle detection and terrain classification for autonomous off-road navigation. *Autonomous Robots*, 18:81–102.

[Manduchi et al., 2001] Manduchi, R., Matthies, L., and Pollara, F. (2001). From cross-country autonomous navigation to intelligent deep space communications: visual sensor processing at jpl. In *Proceedings of the 11th Int. Conf. on Image Analysis and Processing*, pages 472–477.

[Matthies et al., 2003a] Matthies, L., Bellutta, P., and McHenry, M. (2003a). Detecting water hazards for autonomous off-road navigation. In *Proceedings of SPIE Unmanned Ground Vehicle Technology V*.

[Matthies et al., 2003b] Matthies, L., Bergh, C., Castano, A., Macedo, J., and Manduchi, R. (2003b). Obstacle detection in foliage with ladar and radar. In *Proceedings of the International Symposium of Robotics Research (ISRR)*.

[Matthies et al., 1995] Matthies, L., Kelly, A., Litwin, T., and Tharp, G. (1995). Obstacle detection for unmanned ground vehicles: a progress report. In *Proceedings of IEEE Intelligent Vehicles Conference*, pages 66 – 71.

[Matthies and Rankin, 2003] Matthies, L. and Rankin, A. (2003). Negative obstacle detection by thermal signature. In *Proceedings of the IEEE Conference on Robotics and Automation*.

[Maybeck, 1979a] Maybeck, P. S. (1979a). *Stochastic Models, Estimation, and Control: Volume 1*. Academic Press.

[Maybeck, 1979b] Maybeck, P. S. (1979b). *Stochastic Models, Estimation, and Control: Volume 2*. Academic Press.

[Maybeck, 1994] Maybeck, P. S. (1994). *Stochastic Models, Estimation, and Control: Volume 2*, volume 141 of *Mathematics in Science and Engineering*. Navtech Book & Software Store. Republished by Navtech Book & Software Store.

[Melas and Wilson, 2002] Melas, D. E. and Wilson, S. P. (2002). Double markov random fields and bayesian image segmentation. *IEEE Transactions on Signal Processing*, 50(2):357–365.

[Minka, 1998] Minka, T. P. (1998). From hidden markov models to linear dynamical systems. Technical Report 531, Massachusetts Institute of Technology.

[Murphy, 2002] Murphy, K. (2002). Hidden semi-markov models (hsmm). Technical report, Massachusetts Institute of Technology.

[Nashashibi et al., 1994] Nashashibi, F., Fillatreau, P., Dacre-Wright, B., and Simeon, T. (1994). 3-d autonomous navigation in a natural environment. In *Proceedings of the IEEE Int. Conf. on Robotics and Automation (ICRA 94)*, volume 1, pages 433–439.

[O'Conner et al., 1996] O'Conner, M., Bell, T., Elkaim, G., and Parkinson, B. (1996). Automatic steering of farm vehicles using GPS. In *Proceedings of the International Conference on Precision Agriculture*.

[Ollis and Jochem, 2003] Ollis, M. and Jochem, T. (2003). Structural method for obstacle detection and terrain classification. In *Proceedings of SPIE Unmanned Ground Vehicle Technology V*.

[Ollis and Stentz, 1997] Ollis, M. and Stentz, A. T. (1997). Vision-based perception for an autonomous harvester. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robotic Systems*, volume 3, pages 1838 – 1844.

[Ostendorf et al., 1996] Ostendorf, M., Digalakis, V., and Kimball, O. (1996). From HMMs to segment models: a unified view of stochastic modeling for speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 4:360–378.

[Pilarski et al., 2002] Pilarski, T., Happold, M., Pangels, H., Ollis, M., Fitzpatrick, K., and Stentz, A. (2002). The demeter system for automated harvesting. *Autonomous Robots*.

[Pomerleau, 1995] Pomerleau, D. (1995). Neural network vision for robot driving. In Arbib, M., editor, *The Handbook of Brain Theory and Neural Networks*.

[R. Simmons and Whelan, 1996] R. Simmons, L. Henricksen, L. C. and Whelan, G. (1996). Obstacle avoidance and safeguarding for a lunar rover. In *Proceedings AIAA Forum on Advanced Developments in Space Robotics*.

[Rabiner, 1989] Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

[Reid and Searcy, 1987] Reid, J. F. and Searcy, S. W. (1987). Vision-based guidance of an agricultural tractor. *IEEE Control Systems Magazine*, 7(12):39–43.

[Reid et al., 2000] Reid, J. F., Zhang, Q., Noguchi, N., and Dickson, M. (2000). Agricultural automatic guidance research in north america. *Computers and Electronics in Agriculture*, 25:155–167.

[Robert and Casella, 2004] Robert, C. P. and Casella, G. (2004). *Monte Carlo Statistical Methods*. Springer, 2nd edition.

[Rosenblatt, 1997] Rosenblatt, J. (1997). Damn: A distributed architecture for mobile navigation. *Journal of Experimental and Theoretical Artificial Intelligence*, 9(2/3):339–360.

[Sanchéz and Rodellar, 1996] Sanchéz, J. M. M. and Rodellar, J. (1996). *Adaptive Predictive Control: From the concepts to plant optimization*. Systems and Control Engineering. Prentice Hall.

[Saquib et al., 1998] Saquib, S. S., Bouman, C. A., and Sauer, K. (1998). Ml parameter estimation for markov random fields with applications to bayesian tomography. *IEEE Transactions on Image Processing*, 7(7):1029–1044.

[Schaal and Atkeson, 1994] Schaal, S. and Atkeson, C. G. (1994). Assessing the quality of learned local models. In *Neural Information Processing Systems (NIPS)*, pages 160–167.

[Schaal and Atkeson, 1998] Schaal, S. and Atkeson, C. G. (1998). Constructive incremental learning from only local information. In *Neural Computation*, volume 10, pages 147–184.

[Schneider and Moore, 2000] Schneider, J. and Moore, A. (2000). A locally weighted learning tutorial using vizier 1.0. Technical Report CMU-RI-TR-00-18, Robotics Institute, Carnegie Mellon University.

[Seraji and Bon, 2002] Seraji, H. and Bon, B. (2002). Multi-range traversability indices for terrain-based navigation. In *Proceedings of IEEE Int. Conf. on Robotics and Automation (ICRA 02)*, volume 3, pages 2674 – 2681.

[Seraji and Howard, 2002] Seraji, H. and Howard, A. (2002). Behavior-based robot navigation on challenging terrain: A fuzzy logic approach. *IEEE Transactions on Robotics and Automation*, 18:308–321.

[Seraji et al., 2001] Seraji, H., Howard, A., and Tuustel, E. (2001). Safe navigation on hazardous terrain. In *Proceedings of the 2001 Int. Conf. on Robotics and Automation (2001 ICRA)*, volume 3, pages 3084–3091. IEEE.

[Shiller, 2000] Shiller, Z. (2000). Obstacle traversal for space exploration. In *IEEE Int. Conf. on Robotics and Automation (ICRA 00)*, volume 2, pages 989–994.

[Shiller and Gwo, 1991] Shiller, Z. and Gwo, Y.-R. (1991). Dynamic motion planning of autonomous vehicles. *IEEE Transactions on Robotics and Automation*, 7(2):241–249.

[Simeon and Dacre-Wright, 1993] Simeon, T. and Dacre-Wright, B. (1993). A practical motion planner for all-terrain mobile robots. In *Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS 93)*, volume 2, pages 1357–1363. IEEE.

[Singh and Keller, 1991] Singh, S. and Keller, P. (1991). Obstacle detection for high speed navigation. In *Proceedings of IEEE Int. Conf. on Robotics and Automation (ICRA 91)*, volume 3, pages 2798–2805.

[Singh et al., 2000] Singh, S., Simmons, R., Smith, T., Stentz, A., Verma, V., Yahja, A., and Schwehr, K. (2000). Recent progress in local and global traversability for planetary rovers. In *IEEE Int. Conf. on Robotics and Automation (ICRA 00)*.

[Smith, 1983] Smith, D. M. (1983). Algorithm AS 189: Maximum likelihood estimation of the parameters of the beta binomial distribution. *Applied Statistics*, 32(2):196–204.

[Southall et al., 1999] Southall, B., Hague, T., Marchant, J. A., and Buxton, B. F. (1999). Vision-aided outdoor navigation of an autonomous horticultural vehicle. In *Proceedings of the International Conference on Vision Systems*.

[Stentz, 1994] Stentz, A. (1994). Optimal and efficient path planning for partially-known environments. In *IEEE International Conference on Robotics and Automation*.

[Stentz, 2001] Stentz, A. (2001). Robotic technologies for outdoor industrial vehicles. In *Proceedings of SPIE AeroSense*.

[Stentz et al., 2002a] Stentz, A., Dima, C., Wellington, C., Herman, H., and Stager, D. (2002a). A system for semi-autonomous tractor operations. *Autonomous Robots*, 13(1):87–104.

[Stentz and Hebert, 1995] Stentz, A. and Hebert, M. (1995). A complete navigation system for goal acquisition in unknown environments. In *Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, volume 1, pages 425–432.

[Stentz et al., 2002b] Stentz, A., Kelly, A., Herman, H., Rander, P., Amidi, O., and Mandelbaum, R. (2002b). Integrated air/ground vehicle system for semi-autonomous off-road navigation. In *AUVSI Symposium*.

[Swendsen et al., 1992] Swendsen, R. H., Wang, J.-S., and Ferrenberg, A. M. (1992). New monte carlo methods for improved efficiency of computer simulations in statistical mechanics. In Binder, K., editor, *The Monte Carlo Method in Condensed Matter Physics*, volume 71 of *Topics in Applied Physics*, pages 75–91. Springer-Verlag.

[Talukder et al., 2002a] Talukder, A., Manduchi, R., Castano, R., Owens, K., Matthies, L., Castano, A., and Hogg, R. (2002a). Autonomous terrain characterisation and modelling for dynamic control of unmanned vehicles. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS 02)*, pages 708–713.

[Talukder et al., 2002b] Talukder, A., Manduchi, R., Rankin, A., and Matthies, L. (2002b). Fast and reliable obstacle detection and segmentation for cross-country navigation. In *IEEE Intelligent Vehicles Conference 2002*.

[Tunstel et al., 2001] Tunstel, E., Howard, A., and Seraji, H. (2001). Fuzzy rule-based reasoning for rover safety and survivability. In *Proceedings of the 2001 Int. Conf. on Robotics and Automation (2001 ICRA)*, volume 2, pages 1413–1420. IEEE.

[Ulrich and Nourbakhsh, 2000] Ulrich, I. and Nourbakhsh, I. (2000). Appearance-based obstacle detection with monocular color vision. In *Proceedings of the AAAI National Conference on Artificial Intelligence*.

[Urmson, 2004] Urmson, C. (2004). *Navigation Regimes for Off-Road Autonomy*. PhD thesis, Carnegie Mellon University.

[Urmson et al., 2002] Urmson, C., Dias, M. B., and Simmons, R. (2002). Stereo vision based navigation for sun-synchronous exploration. In *IROS 2002*.

[Vandapel and Hebert, 2004] Vandapel, N. and Hebert, M. (2004). Finding organized structures in 3-D ladar data. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS 04)*.

[Vandapel et al., 2004] Vandapel, N., Hebert, M., Huber, D., and Akapuria, A. (2004). Natural terrain classification using 3-D ladar data. In *Proceedings of IEEE Int. Conf. on Robotics and Automation (ICRA 04)*.

[Vijayakumar et al., 2005] Vijayakumar, S., D'Souza, A., and Schaal, S. (2005). Locally weighted projection regression. *Neural Computation*.

[Vijayakumar and Schaal, 2000] Vijayakumar, S. and Schaal, S. (2000). Locally weighted projection regression : An o(n) algorithm for incremental real time learning in high dimensional spaces. In *Proceedings of the Seventeenth Int. Conf. on Machine Learning (ICML 2000)*, pages 1079–1086.

[Wellington et al., 2005] Wellington, C., Courville, A., and Stentz, A. (2005). Interacting markov random fields for simultaneous terrain modeling and obstacle detection. In *Proceedings of Robotics Science and Systems*.

[Wellington and Stentz, 2003] Wellington, C. and Stentz, A. (2003). Learning predictions of the load-bearing surface for autonomous rough-terrain navigation in vegetations. In *Proceedings of the Int. Conf. on Field and Service Robotics*.

[Wellington and Stentz, 2004] Wellington, C. and Stentz, A. (2004). Online adaptive rough-terrain navigation in vegetation. In *Proceedings of IEEE Int. Conf. on Robotics and Automation (ICRA 04)*.

[Winkler, 2003] Winkler, G. (2003). *Image Analysis, Random Fields, and Markov Chain Monte Carlo Methods*. Springer.

[Yoshida and Hamano, 2002] Yoshida, K. and Hamano, H. (2002). Motion dynamics of a rover with slip-based traction model. In *Proceedings of IEEE Int. Conf. on Robotics and Automation (ICRA 02)*, volume 3, pages 3155 – 3160.

[Zhang et al., 1999] Zhang, Q., J.F.Reid, and Noguchi, N. (1999). Agricultural vehicle navigation using multiple guidance sensors. In *Proceedings of the Int. Conf. on Field and Service Robotics*.

[Zhang et al., 2001] Zhang, Y., Brady, M., and Smith, S. (2001). Segmentation of brain mr images through a hidden markov random field model and the expectation-maximization algorithm. *IEEE Transactions on Medical Imaging*, 20(1):45–57.