Learning Rough-Terrain Autonomous Navigation

J. Andrew Bagnell, David Bradley, David Silver, Boris Sofman
Robotics Institute, Carnegie Mellon University

***Abstract*—Autonomous navigation by a mobile robot through natural, unstructured terrain is one of the premier challenges in field robotics. The DARPA UPI program was tasked with advancing the state of the art in robust autonomous performance through challenging and widely varying environments. In order to accomplish this goal, machine learning techniques were heavily utilized to provide robust and adaptive performance, while simultaneously reducing the required development and deployment time. This paper describes the autonomous system, Crusher, developed for the UPI program, and the learning approaches that aided in its successful performance.**

***Index Terms*—Field robotics, machine learning.**

Fig. 1: Spinner (left) and Crusher (right) robots used throughout the UPI program. The natural terrain shown here is representative of the domains they operate within.

I. Introduction

Autonomous robots capable of navigating in challenging unstructured terrain have many potential commercial, industrial and military applications. Tremendous advances in autonomous navigation have been made recently in field robotics [**?**], [**?**], [1]; machine learning has played an increasingly important role in these advances. The DARPA UPI program was conceived to take a fresh approach to all aspects of autonomous outdoor mobile robot design: from vehicle design to the design of perception and control systems. The essential problem addressed by the UPI program is to enable safe autonomous traverse of a robot from Point A to Point B in the least time possible given a series of waypoints separated by 0.2 km to 2 km.

Development on the UPI program was driven by extensive field testing on unrehearsed terrain to meet key metrics in off-road speed and required human interventions. These tests were conducted in extremely challenging terrains, including everything from temperate forests to high deserts, and required negotiating steep slopes, large boulders, deep washes, and dense vegetation. In the complex terrain considered, it was recognized that some human intervention was inevitable, although performance metrics strictly limited that availability. A more in-depth discussion of the program as a whole, as well as the autonomy components of the program can be found in [**?**], [2].

The UPI program required new approaches to meet program speed and time-between-intervention goals that were approximately an order of magnitude greater than anything autonomous robots had previously achieved in such complex terrain. In particular, the diversity of obstacles, vegetation, and terrain overwhelmed approaches based on hand-engineering all aspects of the system using the traditional algorithmic tools of field robotics. Instead, the UPI team combined proven engineering techniques developed within the field including model-predictive control [**?**], [1], 3D LADAR analysis [1], [4], and fast motion re-planning [5], with data-driven, machine-learning approaches. This synthesis enabled us to "program-by-demonstration" some of the most challenging aspects of the problem, including, for example, the automatic interpretation of ambiguous sensor data, as well as to adapt the system's performance automatically to novel, previously unseen terrain.

Below we lay out the core elements of this approach to mobile robotics and the key role played by machine learning in improving performance, enabling non-experts to modify the system, and leading to dramatic reductions in engineering time and effort. The learning approaches that enabled this performance span a diversity of techniques within the machine learning field including: classification and regression, self-supervised learning, imitation learning, online (no-regret) learning, and iterative learning control. Learning techniques are nearly ubiquitous in our approach, touching the full array of subsystems on the Crusher platform including near-range LADAR analysis, far-range monocular and stereo image interpretation, overhead imagery and height map analysis, local and global motion planning, motor control, and vehicle positioning.

The adaptive techniques presented here, when combined with the non-adaptive algorithmic approaches and inherent vehicle mobility, have arguably achieved the state-of-the-art in outdoor, complex terrain autonomy. We discuss some of the extensive experimental testing and validation of these techniques. We conclude with a discussion of the key lessons learned on the integration of learning into autonomous systems and the many challenges that remain in achieving high-performance autonomy while maximizing the value of collected data and minimizing engineering time and effort.

II. Crusher Autonomous System

The Crusher UGV of the UPI Program (shown in Figure 1 with its predecessor, Spinner) was designed to navigate autonomously through complex, outdoor environments, using a combination of LADAR and camera sensors onboard the vehicle as well as satellite or other prior overhead data when

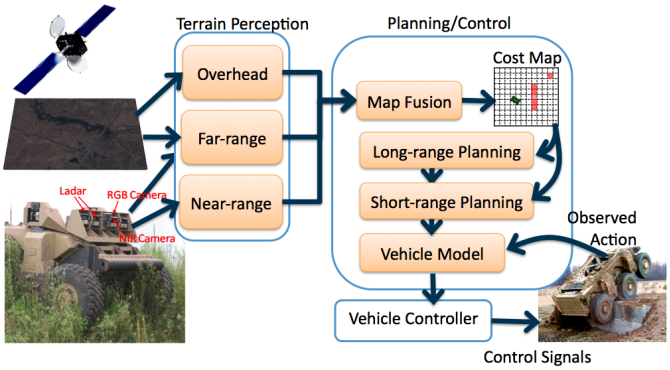


Fig. 2: Crusher's autonomous navigation system combines data from camera and LADAR sensors onboard the robot with previously collected satellite imagery to find safe and efficient paths through complex off-road environments.

available [2]. Figure 2 outlines the high level data flow within the Spinner and Crusher autonomy systems.

Figure 2 hides crucial practical details of the system, including positioning, expiration of data, and inter-process communications and timing, but captures the core elements of the system important for a discussion of the role of machine learning. The various modules and heuristics used in the autonomy system contain a large number of settings and parameters. Manually engineering a good set of parameters for these modules is both difficult and time consuming. A key principle of the approach taken on the UPI program is to learn the parameters of the system from data or human demonstration wherever possible. As will be discussed in the following sections, Crusher's reliable performance was achieved by developing and applying state-of-the-art learning techniques for supervised learning, self-supervised learning, and unsupervised learning through this data-flow architecture. Space limits our ability to discuss all of the learning modules throughout the system, and thus we focus on a few that were especially critical to the results of the overall system.

Crusher's onboard perception system is divided into two modules. A near-range perception system uses a combination of onboard camera and LADAR sensors to produce a continuously updating high resolution model of its environment out to a range of 20 m. These models consist of various geometric or appearance based features assigned to individual 3-D voxels[1]. Supervised learning (Section III-A) is used to classify 3-D voxels into "ground", "obstacle", and "vegetation" classes from a set of labeled terrain examples. In addition, a far-range perception system produces a lower resolution model out to a range of 60 m. Due to the limited sensor resolution and sparse LADAR returns from distant objects, properly interpreting far-range data is quite difficult. Section III-D describes a powerful self-supervised learning technique that was developed to "bootstrap" the far-range perception system using the output of the near-range system. In addition, both the near-range and far-range modules maintain an estimate of the local terrain supporting surface (Section III-B).

A useful and common abstraction for navigation is to represent the world as a 2-D horizontal grid around the robot (see Figure 3) [1]. Navigation through the environment is achieved by first producing a *traversal cost* for driving through each cell in the 2-D grid and then planning the minimum cost path through the grid. These costs are generated as a function of the features associated with each 3-D voxel within the appropriate 2-D column. In practice, it is remarkably difficult to hand-engineer an effective mapping from perceptual features to traversal costs that produces plans and vehicle behaviors that correspond to our expectations– identifying such cost functions from limited and ambiguous perception data is effectively a "black art" for mobile robotics. The UPI program developed novel techniques for imitation learning that allow a cost function to be learned directly from examples of preferred driving behavior (Section IV-A). These costs, along with costs generated from available prior overhead data, are then merged and passed on the Crusher's planning system. The processing of overhead data mirrors that of Crusher's onboard perception system, and is described in Section III-C.

The planning system utilizes a hybrid global/local approach [1] that continuously replans to take into account new information. The long-range global path to the goal is computed by finding a minimum cost path through the map using the Field D* algorithm [5]. The output of the global planner is refined by the local planner in a kinematically constrained search over a small area around the robot, where it is able to plan sophisticated actions and maneuvers that incorporate the robot's mobility and dynamics [?]. Planning such maneuvers requires an accurate model of the vehicle motion produced by a given set of control signals. As Crusher is a 6 wheeled skid-steer vehicle, vehicle motion can be quite difficult to model; therefore self-supervised learning was applied (Section IV-B) to learn the vehicle's response to control inputs and improve its ability to follow commanded paths.

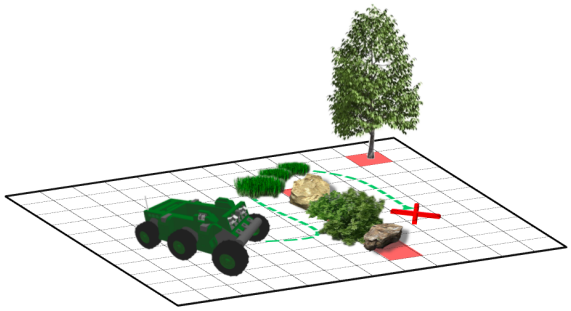


Fig. 3: Abstraction of the mobile robot navigation problem. Sensor data is used to estimate *traversal costs* for each cell in a 2-D grid around the robot. An efficient planning algorithm is then used to find the minimum cost path (green line) to a goal (red X) defined by a set of GPS coordinates.

III. Terrain Perception

Crusher employs learning-based perception modules for spatial regions close to the vehicle, farther from the vehicle, and areas which are outside of the range of the vehicles onboard sensors, but for which there is available satellite or aerial data.

[1] A "voxel" is the 3-D equivalent of a pixel

A. Near-range Terrain Classification

A set of LADAR scanners collect 3-D range data while cross-calibrated color and near infra-red (NIR) cameras determine appearance data from the vehicle's surroundings. Within 20 meters of Crusher, these sensors provide abundant data for terrain analysis. The input to the near-range perception

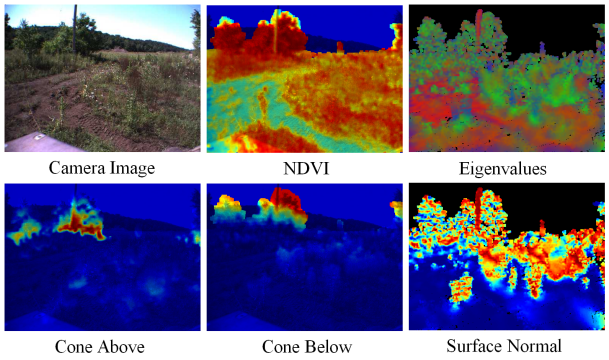


Fig. 4: The camera and laser data captured by the sensors onboard the mobile robot are used to compute a set of engineered features. Here a few of the features are shown projected into the image from one of the onboard cameras.

system consists of 3-D points from the LADAR that have been projected into the camera images and "tagged" with local properties of the image such as color, and image texture. The local perception system discretizes the space surrounding the robot into a 3-D grid of voxels and computes features of the tagged points over each voxel. Examples of some of these engineered features are shown in Figure 4, and they include the averages of the point tags, eigenvalues of the local point cloud scatter matrix [4], the surface normal (3rd eigenvector of the scatter matrix), and the probability and strength of laser pulse reflection from the voxel. A particularly useful feature for vegetation detection was the average Normalized Difference Vegetation Index (NDVI) value of each voxel, which is computed from the red and near-infrared color tags of each point [**?**]. NDVI was developed in the satellite imaging community to exploit the unique spectral properties of small water-filled plants cells that contain chlorophyll. Figure 5 shows vegetation highlighted in bright green that was detected using NDVI on a pair of color and NIR images.

Fig. 5: Left: vegetation detected with the NDVI feature is highlighted in bright green. Right: original color and NIR images. Note how bright vegetation appears in NIR.

The engineered features are then classified with a multi-class logistic regression (MaxEnt) classifier into "ground", "vegetation", or "obstacle" classes, using a labeled data set of over 4 million voxels collected from a variety of test environments. Collection of such a large data set was made feasible by the development of an intuitive image-based voxel labeling tool. In some terrains, the wide variety of examples included in the labeled training set actually hurt classification performance. For example, the NDVI feature improves performance in terrains (and seasons) where chlorophyll-rich vegetation is abundant, but over-dependence on NDVI hurts classification performance in areas with dead or desert-type vegetation. To fix this problem, a method was developed [17] to automatically adapt the terrain classifier to the environment the robot is currently operating in. The unlabeled voxels the robot is trying to classify are used to give more weight to similar examples from the labeled data set, and the classifier is retrained on the reweighted data set.

Figure 6 visualizes the output of the near-range classification system on a forest scene in Colorado. The terrain classification of each voxel is combined with a ground height or terrain surface estimate for each 2-D cell to produce a traversal cost estimate for each voxel, using the imitation learning techniques described in Section IV-A.

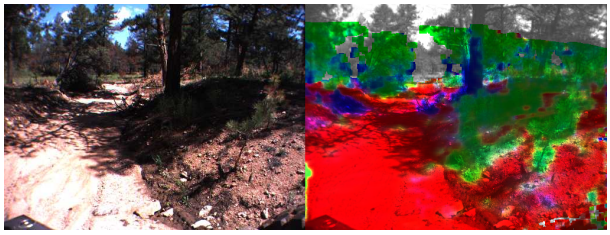


Fig. 6: Classified voxels in a forest scene.

B. Terrain Surface Estimation

A central element of successful rough terrain navigation systems is the estimation of the terrain supporting surface. Identifying the surface enables the autonomy system to detect hazards due to its shape including high grades, ditches, holes or high-centering hazards. Further, identifying obstacles requires knowing where the ground is– a tree limb at the level of the sensors can potentially disable the robot, while at a higher elevation it may not interfere at all.

Unfortunately, identifying the ground surface is complicated by vegetation, occlusion, and sparsity in LADAR point density (see Figure 7). Identifying this surface is perhaps more naturally viewed as an estimation problem then one of machine learning. Indeed, while our approach builds on past work in this area [1], [6], we determined that a method from the online learning community provided a remarkably fast and effective algorithm.

Our approach to estimation leverages the notion, developed in [1], that each LADAR "point" from the sensor is truly a ray from the sensor to a point in the world and that this ray defines a "space-carving" constraint on the ground supporting surface– at any two-dimensional discretized grid location in the world the surface must lie beneath that ray. This approach makes sparse LADAR data dramatically more effective for

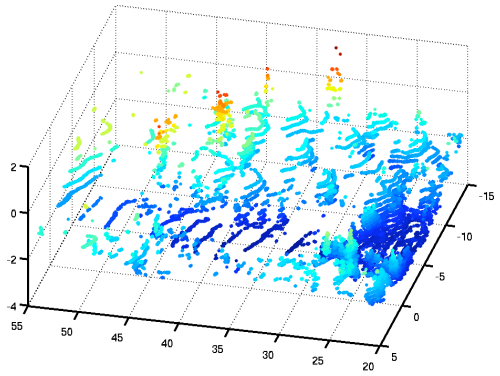


Fig. 7: LADAR scans of rough terrain often have complex structure and highly variable resolution that decays rapidly with distance.

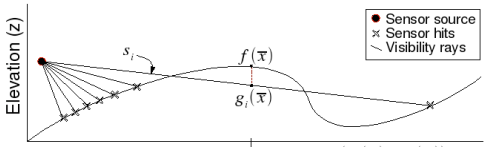


Fig. 8: Constraint on ground supporting surface implied by space-carving rays.

upper bounding the ground at each location than if only the reflection points were considered [7]. In practice, however, this approach remains insufficient in areas where data is especially sparse and particularly where there is vegetation. In [6], the authors develop a Markov Random Field (MRF) based technique over three-dimensional voxels that estimates terrain supporting surface by leveraging contextual information from neighboring cells using a computationally demanding Markov-Chain Monte Carlo procedure.

On the UPI program, these ideas are combined in a two-dimensional MRF formed with a random variable representing height at each location. This random field uses both LADAR points and terrain classification to establish potentials between the random variables– for instance, if terrain classification detects bare earth, the random field strongly enforced the ground height at that location. More complete details of the approach including a detailed description of the random field and its interaction with terrain classification can be found in [?]. The connection with machine learning is perhaps a bit surprising: the fast, anytime estimates of the ground plane necessary for autonomous navigation were made possible by the use of the online projected gradient convex optimization described in [8] to iteratively infer the most probable height of the supporting surface at each 2-D location. At every iteration, the optimization takes a gradient step by moving each cell's ground estimate along the vertical axis towards a position suggested by the neighbors of that cell, effectively anisotropically smoothing the ground heights. Each cell is then projected onto the constraint implied by the LADAR rays; if the estimate goes above the lowest point along any ray that intersects a cell, the estimate is clamped back onto that value (see Figure 8 for a visualization).

Recently the online optimization space-carving notion has been extended by combining with Gaussian process priors to provide a discretization-free estimate of the terrain supporting, again using an effective sub-gradient descent approach [7].

C. Overhead Terrain Perception

When the length of an autonomous traverse greatly exceeds the range of a robot's onboard sensors, prior knowledge of the environment can improve autonomous safety and efficiency by guiding the robot over more easily navigable terrain [9]. The overhead vantage point is an attractive source of such prior knowledge. Satellite imagery can now be commercially purchased at high resolution, and can be augmented with digital elevation maps or aerial LiDAR scans. While the wide variety of available data sources increases the available prior knowledge, the heterogenous nature of the data can make processing and interpretation difficult.

In order to reduce the degree of engineering and human effort that must go into processing overhead data sets from multiple sources, learning techniques were used extensively. At a high level, the processing of overhead data mirrors the approach used for near-range perception (Section III-A). First, the available raw data sets are processed into a set of feature maps (e.g. color, texture, and NDVI for imagery and slope, canopy cover, and other geometric features for LiDAR). Along with these features, human labeling of terrain is used as input into a supervised semantic classification (specifically a convolutional neural network) [9]. This approach has the advantage that it is easily adaptable to different environments without additional feature engineering; it only requires a brief human labeling effort. Additionally, when aerial LiDAR scans are available, a terrain surface estimation procedure is performed, based on the same basic algorithm as Section III-B. As with near-range perception, terrain classification is a useful but not necessary input to this procedure. A notable difference is that the sensor origin is often not provided with commercial aerial LADAR; however the generally low angle of incidence means the location of each point in space is a sufficient constraint. After processing, refined overhead data (visual and geometric features plus semantic classifications) are converted into traversal costs to be fused with onboard data; the learning procedure for this conversion is described in Section IV-A.

D. Bootstrapped Far-range Terrain Perception

Roboticists often equip UGVs with powerful sensors and data sources to deal with uncertainty, only to discover that the UGVs are able to make only minimal use of this data and still find themselves in trouble. As described previously, overhead data has the potential to greatly enhance autonomous robot navigation in complex outdoor environments. Likewise, the ability to interpret terrain well at a far distance from onboard sensor data can help the robot find out about the world as early as possible, allowing it to construct more globally efficient paths and reduce risk. In practice, reliable and effective automated interpretation of such data from diverse terrain, environmental conditions, and sensor varieties proves to be challenging.

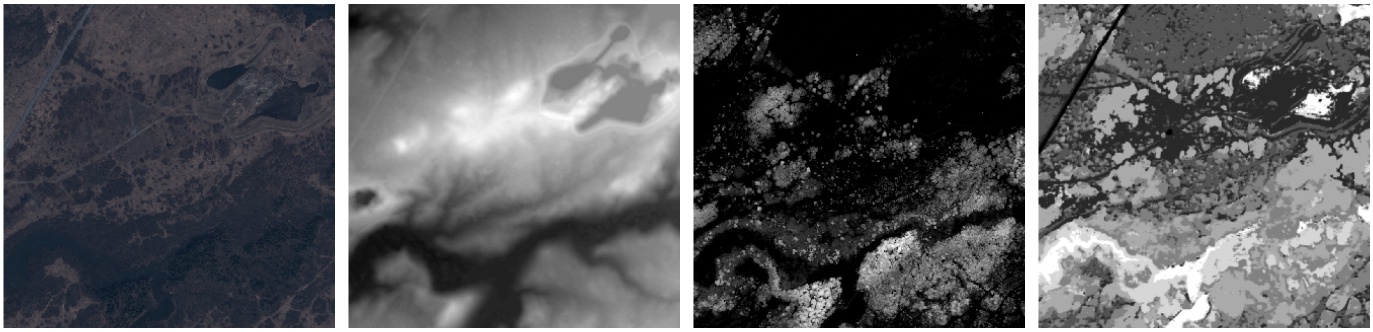


Fig. 9: Examples of overhead data over a 1 km^2 area. From left to right: satellite imagery, extraction ground surface height, tree canopy, semantic classification. Quickbird satellite imagery courtesy of Digital Globe, Inc.

As a result, a system that needs to perform reliably across many domains without re-engineering by hand or supervised training must rely on only the subset of available information that generalizes well across many domains. Due to the data accuracy, consistency and density required for such features, this often limits a system to utilizing only onboard sensor data within a short proximity to the vehicle.

The UPI program employed a bootstrapping system in order to be able to take advantage of all potentially useful data sources. In [10] we introduced a Bayesian probabilistic framework for learning and inferring between heterogeneous data sources that vary in density, accuracy and scope of influence. Here we present a specific instance of this framework that takes advantage of an accurate and robust near-range perception system to extend the perception range of the robot by interpreting any combination of overhead and far-range sensor data with online, self-supervised learning. When dealing with overhead data, our algorithm will be referred to as MOLL (Map On-Line Learning) and when dealing with far range sensor data, our algorithm will be referred to as FROLL (Far-Range On-Line Learning).

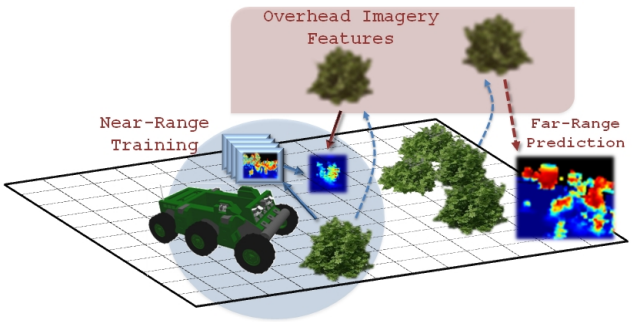


Fig. 10: Abstraction of the online bootstrapped far-range perception system. The near-range perception system generates traversal cost estimates in proximity to the robot (blue region) that are used to learn the mapping from difficult to interpret locale-specific features to traversal cost. This learned model can then be applied elsewhere to produce traversal cost predictions where no near-range perception estimates are available.

The intuition behind this approach is visualized in the scenario shown in Figure 10. Our goal is to produce traversal cost estimates for the environment we are operating in. Throughout the robot's traverse, it observes a sequence of features generated from overhead data and far-range sensor data (we'll call these our locale-specific features, $\mathbf{x}$). Initially, it does not know how to interpret these features.

The robot also observes a sequence of traversal cost estimates, $\tilde{c}$, generated from the near-range perception system for a small subset of the environment that we assume are the true traversal costs, c, with some amount of uncertainty. These estimates are gathered online and are only available for a small subset of the locations in the world while the locale-specific features are available for many more locations.

By remembering the locale-specific features, $\mathbf{x}$, seen during traversal, the subset of these locations for which the near-range perception system has generated estimates, $\tilde{c}$, are used to learn a mapping[2] from $\mathbf{x}$ to c. As the robot traverses through the environment, it refines this model online[3].

We choose a simple model for traversal costs given locale-specific features which models the distribution of the traversal costs as a Gaussian with constant variance and a mean produced by a linear function of the features. This allows us to develop the inference for this linear-Gaussian model in an online fashion by revising the posterior distribution of the model in light of a new Gaussian likelihood that takes into account noise from all features.

This approach offers many advantages including reversible learning (useful when multiple traversal cost estimates of different quality become available for a single location), feature selection, and confidence-rated prediction.

While the system's near-range perception system was effective in a majority of situations, the extended perception range resulted in significantly faster and safer navigation (see Section V). The successful use of this system removed any necessity for engineering an independent far range perception system.

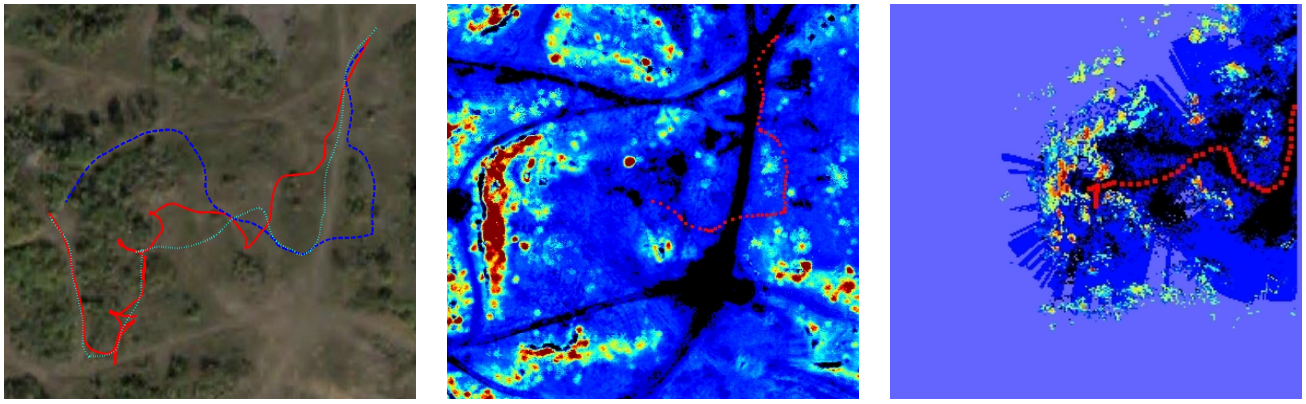
IV. Vehicle Planning and Control

A. Coupling Perception and Planning

The coupling of Crusher's perception and planning systems is achieved through the notion of traversal or mobility costs. Scalar cost values assigned to patches of terrain concretely

[2]Since we are using a linear model, it is more fitting to learn to predict $log(c)$ rather than c which is exponential in nature.

[3]We found that accuracy can be increased by utilizing a layered approach where we learn online to classify locations into ground and non-ground objects and then learn a separate model for each category.



(a) (b) (c)

Fig. 11: Comparison of paths executed for shown situations when using only on-board perception (in solid red) and with MOLL (in dashed blue) and FROLL (in dotted cyan) are shown in (a). In (a) the course started at the top right and ended at the left. Predictions of terrain traversal costs for the environment by our bootstrapping perception system at the times the vehicle chose to avoid the large cul-de-sac in front of it are shown for MOLL in (b) and for FROLL in (c). Traversal costs are color-scaled for improved visibility. Blue and red correspond to lowest and highest traversal cost areas, respectively, with roads appearing in black. In (b) MOLL helped the vehicle avoid the area of dense trees by executing a path that is 43% shorter in 73% less time while in (c) FROLL helped the vehicle identify a lack of opening and avoid traversing deep into dense and hazardous vegetation.

describe the relative preferences between different areas. As the planning system seeks to minimize accrued cost, the cost function implicitly defines Crusher's driving style, *e.g.* how aggressive it will be. A well designed cost function is essential to robust autonomous performance.

Despite the importance of robust cost functions, the task of developing them has previously received little effort. As a result cost function design is often something of a black art, with functions simply manually engineered on a case by case basis. While such manually engineered cost functions can and have produced high performance systems, they often require an enormous investment of human resources to continually design, tune, and validate; further it is often unclear if an optimal coupling has been achieved. Such effort must be re-invested with every modification to the perception or planning subsystems.

While the design of cost functions for both near-range perception and overhead data originally involved manually engineering, later efforts focused on more formal approaches. Specifically, a learning-from-demonstration framework was developed that allowed cost functions to be learned from example vehicle behavior produced from human domain experts[4] This approach is based on the concept of *inverse optimal control* in which the metric for an optimal controller is determined from examples of optimal trajectories. Our learning from demonstration approach is an algorithm known as Learning to Search (LEARCH) [11]. The input to the algorithm is a set of example paths representing the expert's preferred path between a specific start and goal location. LEARCH searches for a cost function that meets the constraint that every example path should be the optimal path between its own start and end location. As there is usually no cost function that will satisfy the constraints of a large set of examples, this search takes the form of a soft constrained optimization, trying to minimize the difference between the cost of each example path and its current corresponding planned path.

This objective function can be minimized by computing the gradient with respect to the costs of individual patches of terrain. As the cost of a path is simply the cost of the terrain it traverses, the local gradient is simply made up of the locations along each path; That is, the objective can be minimized by raising the cost of locations along the current plan, and lowering the cost of locations along the expert example. By using a gradient boosting procedure [12], a generalizable and (potentially) non-linear cost function can be constructed using standard classification or regression techniques. Figure 12 provides a visual example of this procedure in action with overhead data as an input. By accounting for the dynamic and partially unknown nature of onboard perceptual data, a similar procedure can learn a cost function for the near-range perception system from expert tele-operation of Crusher [13]. A cost function is not learned specifically for the far-range perception system; instead the bootstrapping procedure described in Section III-D automatically adapts to the learned near-range function.

B. Vehicle Modeling

An important component of the local planning system is predicting the motion that the vehicle will experience in the next few seconds for a given set of motor commands. Because Crusher is a skid-steered vehicle that has complex interactions with slippery, unknown terrain, it proves difficult to construct a traditional high-fidelity, physics-based vehicle model. When going up hills, the vehicle's wheels often slip, reducing the "along-track" velocity that the vehicle is trying to achieve in its intended direction of travel, and often introducing "cross-track" velocities perpendicular to the vehicle's heading. Turning on slopes is particularly prone to slip, and failure to predict the vehicle's motion correctly can result in collision with obstacles. To account for wheel slip, the UPI system predicts

[4]That is, the expert must be sufficiently familiar with the environment and the vehicle capabilities to produce good example behavior, but need not to be an autonomy expert

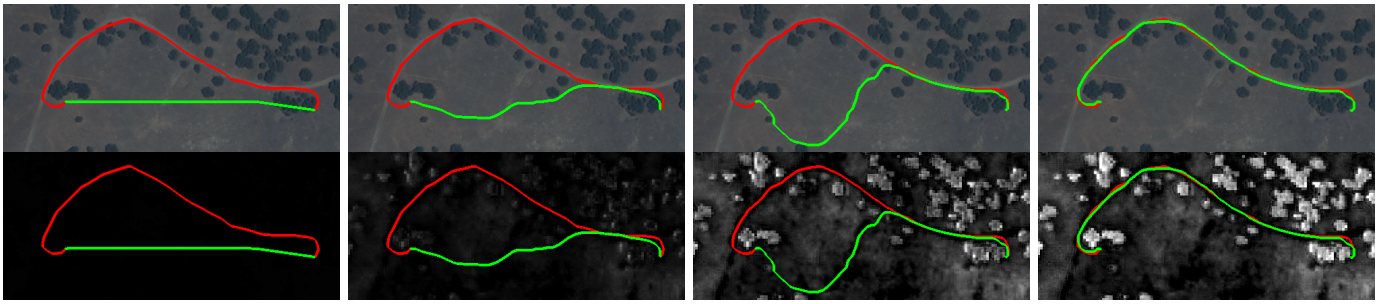


Fig. 12: An example of the LEARCH algorithm learning to interpret satellite imagery (Top) as costs (Bottom). Brighter pixels indicate higher cost. As the cost function evolves (left to right), the current plan (green) recreates more and more of the example plan (red).

the vehicle's motion out to a 3-second time horizon using a 24-layer deep neural network [14]. This neural network predicts the along-track and cross-track velocities the vehicle will experience, as well as the vehicle yaw. The velocity estimates are integrated to produce a predicted path for the vehicle. The current pose (pitch and roll) and velocities provided by the vehicle's IMU, as well as the commands that will be provided to the motors in the 3-second window are used as input. The ground surface estimate discussed in Figure III-B is used to predict the pitch and roll that the vehicle will experience in future locations.

The predictor is trained from logs of the commands sent to the vehicle motors and the actual motion of the vehicle from the inertial navigation system. Such deep neural networks are considered difficult to train because prediction errors often cascade into large output errors and due to difficult "credit assignment" required to determine how the many parameters in initial layers of the network influence the final output. This network was successfully learned by adding "local" information to each hidden layer of the network based on the difference between the predicted and actual velocities of the vehicle at each 0.25s interval, leading to a 59% reduction in error over a previous conventional physics-based vehicle model.

V. Experiments

Over the course of the UPI program, numerous large field tests were conducted at various sites across the continental U.S. These tests were conducted in an unrehearsed manner; there was no prior exposure of either the robot or the autonomy team to the test site, and only minimal time (generally a single day) was provided for any additional training of the system. Despite these constraints, impressive results were achieved. The Crusher and Spinner vehicles autonomously traversed more than 1000 km across difficult terrain, often averaging human intervention as little as once every 20 km.

As the focus of this testing was overall system performance, it was not always possible to quantify the effect of various learning techniques, nor even to attribute a performance improvement to a specific approach[5]. However, for a subset of the

[5]As the motivation behind many of the learning components was the infeasibility of an engineered approach, there are several instances where no engineered module for comparison was ever implemented

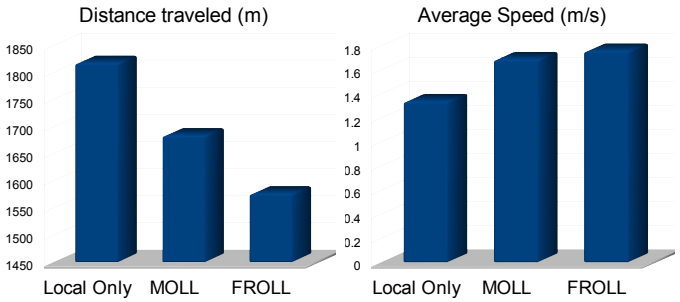


Fig. 13: Over many comparison runs, the bootstrapped far-range perception system was shown to outperform the baseline system using only near-range perception in both distance of travel required and average speed.

implemented learning techniques it was possible to perform specific comparison experiments.

The bootstrapped perception system approach was tested with both overhead data and far-range sensor data to measure its impact on navigation performance compared to a system using only the near-range perception system. The test environments contained a large variety of vegetation, various-sized dirt roads (often leading through narrow passages in dense vegetation), hills, and ditches.

The vehicle traversed several courses defined by a series of waypoints using only its on-board perception system for navigation. It then traversed the same courses with the help of our bootstrapping approach. First, MOLL was applied using 40 cm resolution overhead imagery and elevation data to supplement the on-board perception system with traversal cost updates computed within a 75 m radius once every 2 seconds. Next, FROLL was used to interpret and make predictions from far-range sensor data every 1 second. The algorithm was initialized for each course with no prior training to simulate its introduction into a previously unencountered domain.

Quantitative results can be seen in Figure 13 and a sample scenario where such an approach is especially valuable can be seen in Figure 11. In general, we found that both techniques not only improved the quality of the paths chosen by the vehicle but also allowed higher speed navigation by increasing the time the vehicle had to react to upcoming obstacles and identifying safer terrain such as roads. By learning online to leverage potentially powerful, but difficult to generalize, features from overhead and far-range sensor data, our system is able to use all potentially useful data sources and to adapt to changing conditions without the necessity of human-

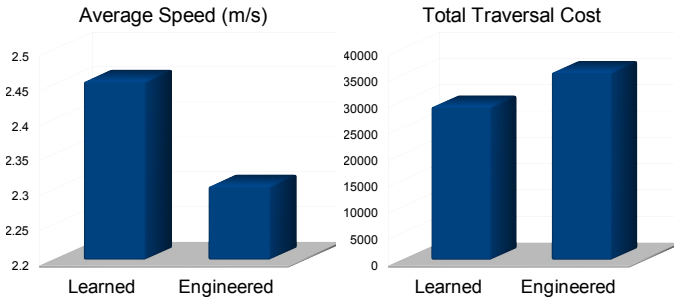


Fig. 14: Comparing performance over an approximately 12 km course, Crusher drover faster and went over safer terrain (as scored independently by onboard perception) when using a learned interpretation of prior data.

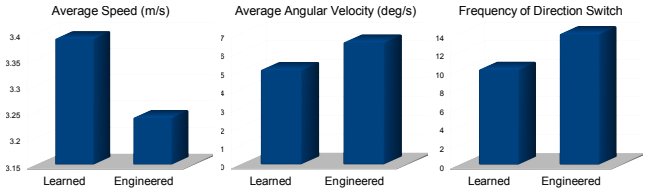


Fig. 15: Comparing performance over approximately 35 km of autonomous traverse, Crusher drove faster, turned softer, and changed direction yes when using a learned cost function. In addition, there was not statistically significant difference in safety between the two systems [15].

supervised retraining or engineering. Additional results and applications are presented in [10].

At the beginning of the UPI program, manually engineered and tuned cost functions were used for both near-range perception and overhead data. Creating cost functions for overhead data was especially time consuming; as each test site consisted of data sets of varying source and quality, a new cost function was generally required for each site. Creating and validating these cost functions usually involved several days of an engineer's time. Midway through the program, this process was changed to make use of the learning from demonstration approach of Section IV-A. In contrast, the learning approach would only require several hours of expert interaction to produce a training set, and resulted in cost functions that produced more desirable long range plans [15]. In addition, the online performance of the system improved; Crusher was shown to drive faster and through safer terrain when using a learned interpretation of overhead data than when using an engineered interpretation (see Figure 14) [16].

Throughout a majority of the UPI program, a manually engineered cost function was used for near-range perception. While this resulted in a high performance system, it came at a high cost, requiring hundreds of engineer-hours of development spread over a three year period. Once an effort was undertaken to apply learning from demonstration, another significant time savings was realized (even though training the system in this context required an expert to actually teleoperate Crusher). The final training set used to train the system consisted of less than a full day of examples (collected piece by piece over different terrains). A large set of comparison experiments demonstrated the learned interpretation of near-range perceptual data to provide an equivalent level of safety to the engineered one, while slightly increasing driving efficiency through faster motion and fewer turns (see Figure 15) [15].

VI. Future of Machine Learning in Mobile Robotics

Despite the impressive performance of the Crusher robot, a crucial role remains for machine learning to improve the performance and reduce the cost of developing autonomous robotic systems. Conceptually, creating an autonomy system involves three steps: defining the problem to be solved, designing the architecture of the solution, and selecting good parameters for that architecture. Defining the problem requires translating performance goals for the robot into a precise mathematical scoring function that provide a scalar evaluation of the performance of a particular system. For instance, the imitation learning approach used by the planning system converts the high-level autonomous navigation goal of driving safely and efficiently through a cluttered off-road environment to a mathematical function that penalizes the planning system for each example of expert human driving that it does not consider to be "optimal". Designing the architecture of the system involves selecting the format of the input and output data, and the general mathematical functions that will be used to map inputs to outputs. Then parameters for each function in the system are selected in order to maximize the scoring function.

Currently, intermediate modules in Crusher's perception system, such as terrain classification, are not directly trained to improve the final scoring function of the system, but rather have been trained to solve tasks, i.e. classification of voxels as "road", "obstacle", or "vegetation", which we believe will produce effective intermediate representations for the overall system. It is unclear how to define those terrain classes for any particular vehicle or even the best number of terrain classes. Worse still, improved performance on an intermediate task does not always translate into improved navigational performance; the metrics on performance are too weakly coupled. Instead, current work [17] is applying methods from the emerging field of *deep learning* [18]–[20] to learn parameters for intermediate modules that directly optimize the final output of the system, i.e. providing "global coordination" of the entire system. The global coordination problem is hard to optimize however, as there are many local maxima, so another promising technique, *multi-task learning* [21], [22], finds related problems that provide cheap and abundant training data, and learns from them a prior belief about which parameter values will work well for each module.

Ultimately the goal of this learning approach is to use human experts where they are most effective: to provide an overall architecture for the solution and to define the problem in intuitive but mathematically precise terms, and manage the "detail" work of tuning this architecture with automated machine learning algorithms.

Additionally, transitioning mobile robotic systems to real-world use requires an extraordinary degree of reliability. Small problems resulting in mildly sub-optimal performance are often tolerable, but major failures resulting in vehicle loss or compromised human safety are not. Given the cost of such

systems and the importance of safety and reliability in many of the tasks that they are intended for, even a relatively rare rate of failure is unacceptable.

Roboticists develop and train their robots in areas representative of those they are expected to operate in, only to discover that they encounter new situations that are not in their experience base that can lead to unexpectedly poor performance or even complete failures. Since it is impossible to prepare for the unexpected, one must assume that such situations will arise during real-world operation. To mitigate this risk a UGV must be able to identify situations that it is likely untrained to handle *before* it experiences a major failure.

This problem can be framed as *novelty detection*: identifying when perception system inputs differ dramatically from prior inputs seen during training or previous operation. We have achieved initial results in exploring online algorithms that can compactly represent the past experiences of the robot in order to identify potentially dangerous situations such as those shown in Figure 16 [23]. With this ability, the system can either avoid novel locations to minimize risk or stop and enlist human help via supervisory control or tele-operation.

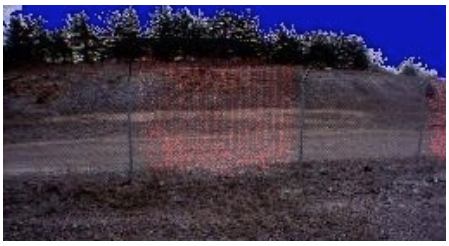


Fig. 16: It is important to identify objects that are not in the systems experience since their interpretation in this case is highly unpredictable. For example, the chain-link fence identified here as novel has characteristics of sparse vegetation in many of its features and would therefore get assigned a dangerously low traversal cost.

Finally, we concede that the complexity and unpredictability of the real world forces robotic systems to adapt online. The key is to identify the feedback that allows online training in as many ways as is feasible, whether it is one part of the system serving as an expert to train another or a human operator serving as the expert at opportune times. We contend that fully utilizing such techniques will enable robots to adapt to and improve their performance in diverse environments with minimal human involvement and greatly expand the effectiveness and potential real-world applications of mobile robotics.

REFERENCES

[1] A. Kelly, A. Stentz, O. Amidi *et al.*, "Toward reliable off road autonomous vehicles operating in challenging environments," *International Journal of Robotics Research*, vol. 25, no. 5-6, pp. 449–483, 2006.

[2] S. Thrun, M. Montemerlo, H. Dahlkamp *et al.*, "Stanley: The robot that won the DARPA grand challenge," *Journal of Field Robotics*, vol. 23, no. 9, pp. 661–692, June 2006.

[3] C. Urmson, J. Anhalt, D. Bagnell *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.

[4] A. Stentz, "Autonomous navigation for complex terrain," Carnegie Mellon Robotics Institute Technical Report, manuscript in preparation.

[5] A. Stentz, J. Bares, T. Pilarski, and D. Stager, "The crusher system for autonomous navigation," in *AUVSIs Unmanned Systems*, August 2007.

[6] T. Howard and A. Kelly, "Optimal rough terrain trajectory generation for wheeled mobile robots," *International Journal of Robotics Research*, vol. 26, no. 2, pp. 141–166, 2007.

[7] N. Vandapel, D. Huber, A. Kapuria, and M. Hebert, "Natural terrain classification using 3-d ladar data," in *Proceedings of the IEEE International Conference on Robotics and Automation*, April 2004.

[8] D. Ferguson and A. Stentz, "Using interpolation to improve path planning: The field D* algorithm," *Journal of Field Robotics*, vol. 23, no. 2, pp. 79–101, February 2006.

[9] D. Bradley, R. Unnikrishnan, and J. Bagnell, "Vegetation detection for driving in complex environments," in *IEEE International Conference on Robotics and Automation*, April 2007.

[10] D. M. Bradley, "Learning in modular systems," Ph.D. dissertation, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, August 2009.

[11] C. Wellington, A. Courville, and A. Stentz, "Interacting markov random fields for simultaneous terrain modeling and obstacle detection," in *Proceedings of Robotics Science and Systems*, June 2005.

[12] R. Hadsell, J. A. Bagnell, and M. Hebert, "Accurate rough terrain estimation with space-carving kernels," in *Proc. Robotics Science and Systems*, June 2009.

[13] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," in *ICML*, 2003.

[14] D. Silver, B. Sofman, N. Vandapel, J. A. Bagnell, and A. Stentz, "Experimental analysis of overhead data processing to support long range navigation," in *Proceedings of the IEEE/JRS International Conference on Intelligent Robots and Systems*, October 2006.

[15] B. Sofman, E. L. Ratliff, J. A. Bagnell, J. Cole, N. Vandapel, and A. Stentz, "Improving robot navigation through self-supervised online learning," *Journal of Field Robotics*, vol. 23, no. 12, December 2006.

[16] N. D. Ratliff, D. Silver, and J. A. Bagnell, "Learning to search: Functional gradient techniques for imitation learning," *Autonomous Robots*, vol. 27, no. 1, July 2009.

[17] L. Mason, J. Baxter, P. Bartlett, and M. Frean, "Boosting algorithms as gradient descent," in *Advances in Neural Information Processing Systems 12*. Cambridge, MA: MIT Press, 2000.

[18] D. Silver, J. A. Bagnell, and A. Stentz, "Perceptual interpretation for autonomous navigation through dynamic imitation learning," in *International Symposium on Robotics Research*, August 2009.

[19] M. W. Bode, "Learning the forward predictive model for an off-road skid-steer vehicle," Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-07-32, September 2007.

[20] D. Silver, J. A. Bagnell, and A. Stentz, "Applied imitation learning for autonomous navigation in complex natural terrain," in *Field and Service Robotics*, July 2009.

[21] ——, "High performance outdoor navigation from overhead data using imitation learning," in *Proceedings of Robotics Science and Systems*, June 2008.

[22] Y. Bengio, "Learning deep architectures for AI," *Foundations and Trends in Machine Learning*, vol. to appear, 2009.

[23] Y. LeCun, U. Muller, J. Ben, E. Cosatto, and B. Flepp, "Off-road obstacle avoidance through end-to-end learning," in *Advances in Neural Information Processing Systems (NIPS 2005)*. MIT Press, 2005.

[24] R. Hadsell, P. Sermanet, M. Scoffier *et al.*, "Learning long-range vision for autonomous off-road driving," *Journal of Field Robotics*, vol. 26, no. 2, pp. 120–144, February 2009.

[25] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," *Proceedings of the 25th international conference on Machine learning*, pp. 160–167, 2008.

[26] R. K. Ando and T. Zhang, "A framework for learning predictive structures from multiple tasks and unlabeled data," *Journal of Machine Learning Research*, pp. 1817–1853, November 2005.

[27] B. Sofman, J. A. Bagnell, and A. Stentz, "Anytime online novelty detection for vehicle safeguarding," Robotics Institute, Carnegie Mellon University, Tech. Rep., April 2009.