Online Learning Techniques for Improving Robot Navigation in Unfamiliar Domains

Boris Sofman

CMU-RI-TR-10-43

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

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

December 2010

Thesis Committee:
Tony Stentz, co-chair
J. Andrew Bagnell, co-chair
Christopher Urmson
Lawrence Jackel, AT&T Labs Division Manager (Emeritus)

Keywords: Mobile robots, field robotics, robot perception, overhead data interpretation, online learning, novelty detection, change detection, candidate selection, list maintenance

Abstract

Many mobile robot applications require robots to act safely and intelligently in complex unfamiliar environments with little structure and limited or unavailable human supervision. As a robot is forced to operate in an environment that it was not engineered or trained for, various aspects of its performance will inevitably degrade. Roboticists equip robots with powerful sensors and data sources to deal with uncertainty, only to discover that the robots are able to make only minimal use of this data and still find themselves in trouble. Similarly, roboticists develop and train their robots in representative areas, only to discover that they encounter new situations that are not in their experience base. Small problems resulting in mildly sub-optimal performance are often tolerable, but major failures resulting in vehicle loss or compromised human safety are not.

This thesis presents a series of online algorithms to enable a mobile robot to better deal with uncertainty in unfamiliar domains in order to improve its navigational abilities, better utilize available data and resources and reduce risk to the vehicle. We validate these algorithms through extensive testing onboard large mobile robot systems and argue how such approaches can increase the reliability and robustness of mobile robots, bringing them closer to the capabilities required for many real-world applications.

Funding Sources

We gratefully acknowledge the sponsors of this research, without whom this thesis would not be possible:

This work was partially sponsored by DARPA under contract Unmanned Ground Combat Vehicle - PerceptOR Integration (contract number MDA972-01-9-0005) and by the U.S. Army Research Laboratory under contract Robotics Collaborative Technology Alliance (contract number DAAD19-01-2-0012).

I have also been partially supported for most of my PhD by a Sandia National Laboratories Excellence in Engineering Fellowship. I would like to gratefully acknowledges Sandia's Campus Executive Laboratory Directed Research and Development Program (LDRD) for this support. I truly appreciate the freedom and flexibility that this generous fellowship has given me.

Acknowledgments

There are many to thank for this dissertation, but I must start with my advisors, Tony and Drew. They have been a source of endless support, insight, guidance and patience, putting up with my changing interests, and even a nine month leave from the program midway through to chase an exciting opportunity in California. I have tremendous respect for both of you and hope that we will stay part of each other's lives. Thank you also to my committee members, Chris Urmson and Larry Jackel, for the insightful discussions and suggestions for this work.

The projects I have had the pleasure of working on would not have been possible without a large number of people behind the scenes. Thank you to Dave S., Cliff, Dave B., Cris, Carl, Mike, Tom, Constantine, and the entire UPI project team who came together on a truly special project. Many thanks also go out to John Cole, Nicolas Vandapel and Ellie Lin with whom I collaborated heavily on the earlier portions of this work.

Switching onto the CTA project for my later work was special because I'd started my robotics career at Carnegie Mellon on this project as an undergraduate, and now I returned to it to conclude my PhD. Thanks to everyone involved, both past (Marc, Dave, Nidhi, Rob, Juan Pablo) and present (Dave, Dom, Balajee, Bernadine, Freddie, Jimmy, Nisarg). I want to specifically thank Brad Neuman, the super-undergrad that everyone looks for but seldom finds. He's been a great partner throughout the past year and his contribution to the later change detection research has been enormous. Good luck in starting your own PhD journey this year.

Carnegie Mellon has been a special place to me for a long time, and one of the main reasons has been a wonderful collection of friends and colleagues. You are too numerous to mention but without you my time here would not have been nearly as enjoyable. You know who you are: thank you.

And finally, I want to give special thanks to my family. My parents, Lev and Anna, for pushing me in the right directions in life, my sister Marianna, who saw my enthusiasm for Carnegie Mellon and is now finishing her undergraduate here, and of course Dana, my wife and best friend. Dana, you've celebrated the good times with me and helped me through the tough times. We've been through a lot these past few years, and I'm sure we'll have plenty more adventures together in the future.

Finally, thank you to Carnegie Mellon. This is a wonderful place to be a student. I've learned more than I could have imagined and have met some incredible people along the way. Thanks for the best lesson of all:


DON'T EVER GIVE UP!

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

The vision of autonomous mobile robots revolutionizing the way we live our lives dates back almost a century. It is easy to justify their appeal: robots could automate many of the tasks that are too unappealing or dangerous to be suitable for humans and handle others at which they are simply more productive and accurate than humans can ever hope to be. These systems could automate tasks in diverse domains such as transportation, military reconnaissance and supply routes, agricultural tasks, space exploration, border and property patrolling and working in toxic environments.

The current state of autonomous robot navigation, however, has yet to fulfill these expectations. Almost all real-world uses of unmanned ground vehicles (UGVs) operate either in highly structured or controlled environments, where the state of the world is fully known at all times and plans can be precisely executed with minimal uncertainty or risk, or with extensive human involvement (see Figure 1.1).

Figure 1.1: Example real-world applications of mobile robotics: from left, the Kiva Systems inventory management robot, the KUKA industrial robot and the iRobot Packbot used for remote bomb disposal. As with most current real-world applications of robotics, these systems do not have to deal with complicated robot perception problems since they operate in controlled and structured environments or through tele-operation.

In reality, for mobile robots to be truly useful, they need to be robust enough to be able to sense and operate in partially unknown and unstructured environments. While many autonomous UGVs have advanced to a level where they are competent and reliable a high percentage of the time in many environments [12, 40, 52, 59, 129, 135], most of these systems are heavily engineered for the domains they are intended to operate in. Any deviation from these domains

often results in sub-optimal performance or even complete failure. 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. In many domains that are prime candidates for mobile robotic applications, the risk of catastrophic failure, however small, is a primary reason why autonomous systems are still under-utilized despite already demonstrating impressive abilities.

A majority of the effort to date in applying machine learning to robotics has been focused on offline learning, either in processing data prior to or after a traversal or in training portions of an autonomy system prior to operation. Such efforts have resulted in a great deal of progress, but these approaches are simply not well suited for dealing with some of the challenges a robot will face. While roboticists do everything they can to equip robots with capabilities relevant to a wide range of domains, in order for real-world applications of UGVs to increase, they must be able to adapt to changing and unfamiliar aspects of their environments. The uncertainty throughout the UGV development and deployment process can lead to degraded performance through several challenging circumstances:

Figure 1.2: UGVs navigating with a limited perception range will often execute highly suboptimal paths. The path above was executed by a large UGV whose goal was to navigate to the goal at the left portion of the environment using an onboard perception system with a range of approximately 15 meters. Due to its limited visibility range, the UGV lost significant time navigating through the heavy vegetation while taking heavy risks in the process that necessitated a human intervention at one point.

Developmental Uncertainty. Roboticists 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. Overhead imagery data, for instance, has the potential to greatly enhance autonomous robot navigation in complex outdoor environments. In practice, reliable and effective automated interpretation of imagery from diverse terrain, environmental conditions, and sensor varieties proves to be challenging. As a result, a system that needs to per-

form reliably across many domains without major re-engineering 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. This can lead to highly suboptimal behavior, often putting the UGV in dangerous situations such as the one shown in Figure 1.2.

In many difficult domains it is important to be able to interpret terrain well at a distance. Finding out about world as early as possible allows a UGV to construct more globally efficient paths while reducing risk. It is therefore important for a system to be able to take advantage of all potentially useful data sources and extend the range of its perception system by adapting to changing conditions without the necessity of human-supervised retraining.

Deployment Uncertainty. Similarly, roboticists develop and train their robots in representative areas, only to discover that they encounter new situations that are not in their experience base. If the perception system does not extend well to situations that were not represented during training, encounters with novel situations 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 therefore becomes one of novelty detection: how a robot can identify when perception system inputs differ from prior inputs seen during training or previous operation in the same area. 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. For maximal impact such a system must be well-suited for online use as the system must incorporate feedback received throughout its continued progress and human aid.

Autonomy system limitations. At times, even the most robust autonomous systems are simply unfit to handle a given situation. Fortunately, in many cases an autonomous UGV is able to seek occasional help from a remote operator. The key is knowing *when* to ask for this help as human time is often valuable and limited. If we treat the human as an oracle with a high query cost, reasoning about the possible impact of human involvement in a situation can avoid a large number of unnecessary human queries. Since novel situations pose the greatest risk, if resolving such uncertainty through human help has a high potential benefit, a system can stop and call for remote help based on the assumption that humans are better able to negotiate new situations than autonomous vehicles. Furthermore, a learning system can observe the performance of the autonomous vehicle in particular situations and compare that performance to remote human-control performance in similar situations. When the vehicle encounters such situations in the future, it can then invoke whichever expert demonstrated better performance: the remote human or autonomous vehicle. When used in conjunction with a novelty detection system, such an approach can safeguard a UGV while maximizing the benefit from human availability throughout autonomous navigation.

Throughout this thesis, we address each of these challenges by presenting a series of online approaches that allow a UGV to adapt to the uncertainty of unfamiliar domains where human aid is limited or unavailable. For each problem we present relevant existing techniques followed by our approaches and an examination of how they address the limitations of similar techniques.

We implement and test these algorithms on large outdoor mobile robots and argue how the added ability to navigate safely and reliably in diverse real-world environments may facilitate the deployment of such robotic systems years earlier than otherwise possible.

1.1 Operating Under Uncertainty

Operating under uncertainty is a central requirement of most non-trivial mobile robot tasks and a central focus of this thesis. The obvious initial step for dealing with such uncertainty is with sensors and a perception system to interpret the incoming sensor data. These perception systems are often trained extensively on data representative of the environment the robot is to operate in, allowing them to parse meaningful information from this sensor data and identify degrees of traversability for the environment. A variety of sophisticated planning algorithms such as Field D* [33] are adept at handling streams of traversal information updates and modifying paths appropriately in real-time.

In existing commercial robots, sensor systems can range from the simple bump and react mechanism of the iRobot Roomba vacuum cleaner to the much more sophisticated line of industrial material handling robots from Seagrid Corporation that construct and operate on three-dimensional occupancy maps. Within the research domain, such examples are varied and numerous, including the systems mentioned earlier [12, 40, 52, 59, 129, 135].

All of these systems make assumptions about the types of environments they need to operate within. The limitation of such robots thus becomes their dependency on their pre-trained perception systems. While they often allow robots to operate within varieties of controlled or predictable environments, such techniques only work for limited ranges from the robot. Additionally, these systems will inevitably struggle when the environment sufficiently changes. Since robot behavior can become unpredictable, in many applications it is critical to detect such situations and apply vehicle safeguarding techniques.

With current techniques there is no sure way for a vehicle to autonomously deal with all situations it may encounter using only a pre-trained perception system. One way to address this problem is to revert to tele-operation for some or all of the mission. Full tele-operation is prohibitively expensive for many applications due to the degree of required human attention and communications bandwidth. In cases where tele-operation is employed a portion of the time, a policy must determine under which conditions the robot or the human are to take control. Just as in the medical domain, false positives are preferable to false negatives due to the potentially high cost of mistakes. Stentz et al. rely on this idea in their system of semi-autonomous tractors [122]. These tractors would execute a specified task and stop when they detect that something *may* be an obstacle and send an image over a wireless link to a human supervisor. The human can then either attend to the problem or tell the robot to proceed if the obstacle was a false alarm. This allows the system to function autonomously a large portion of the time while remaining cautions and allowing a human to quickly intervene in questionable situations. Since the time required to verify an obstacle is minimal, this allows a single operator to oversee several machines. Effective safeguarding techniques are one of the keys to applying robotic technologies to various outdoor industrial tasks [120].

An extensive discussion on related work in hybrid tele-operation control schemes is presented in section 6.1. As discussed in that section, many systems are designed to ask for help only *after* they find themselves in trouble. In many domains, robots do not have this luxury: a significant mistake may result in the end of the mission or the loss of lives. We argue that online learning techniques are the correct way to deal with such situations and present a series of algorithms to deal with the difficulties described previously.

Because this thesis spans several broad areas of robot autonomy, we reserve a full discussion of each area's related work for their respective chapters.

Chapter 2

Thesis Statement

In this thesis we consider the problem of a highly robust autonomous mobile robot whose mission is to navigate through complex, natural terrain to a specified destination. We assume that this terrain does not have a known structure and is not engineered to support navigation as in the case of highway or urban driving and that all obstacles can be treated as static. Since this terrain is either previously untraversed or has potentially changed since the last traversal, the robot is equipped with a variety of near and far range laser and camera based onboard sensors to perceive its environment, as well as the possible availability of overhead imagery and elevation data. We assume the robot's location is known to within several meters of its true position throughout navigation.

The robot's perception system is able to take advantage of ample computing resources to interpret the terrain through sophisticated algorithms that can be trained prior to this navigation on other terrain. However, there is no guarantee that training terrain will be representative of terrain encountered during a mission, even though there is an expectation of strong overlap.

We assume, however, that a non-expert human may be available for a nominal percentage of operation time *during* navigation to provide remote tele-operation support at the request of the robot. Other than this limited remote assistance, the robot must deal on its own with any environmental, seasonal or temporal differences that complicate its mission.

A combination of three metrics of performance are reasonable in such a domain:

Safety. The safety of the vehicle is of highest priority. At all times during live testing, a human operator supervises the robot behavior and is ready to intervene if the vehicle poses a risk to itself or its environment. The occurrences of such human interventions are tracked and are viewed *extremely* unfavorably since if the situations occurred during actual operation, there would be a high possibility of critical failure or unacceptable environmental impact.

Human assistance required. Human time is considered expensive and therefore should be limited to as small of a portion of total operation time as possible.

Time and distance traveled. As with most navigation tasks, we want to reach our destination while minimizing the overall time and distance of travel.

The approaches we present in this thesis are intended to allow a mobile robot to deal with the

described domain while improving its performance under the above-mentioned metrics.

Chapter 3

Technical Approach

3.1 High-Level Approach

We deal with the common scenario where a UGV must act safely and intelligently in a complex natural environment with little structure and limited or unavailable human involvement. As a UGV's autonomy system is forced to operate in an environment that it was not engineered or trained for, various aspects of its performance will inevitably degrade. We present a series of online algorithms to deal with the challenges discussed earlier and enable a UGV to better deal with uncertainty in order to improve performance and reduce risk to the vehicle.

We begin in Section 4 by proposing an online, probabilistic model to effectively learn to use potentially powerful but domain specific features by leveraging other features that, while perhaps otherwise more limited, generalize reliably. This technique allows the system to capture maximal benefit from common data sources such as overhead imagery without the need for human training as in other overhead interpretation techniques (sample results are shown in Figure 3.1). This efficient, self-supervised learning method allows a UGV to significantly extend the effective range of its onboard perception system in unfamiliar domains.

Small problems resulting in mildly sub-optimal performance are often tolerable, but major failures resulting in vehicle loss or compromised human safety are not. In Section 5 we consider the problems of online novelty and change detection for situations where even a well-adapting near-range perception system is potentially unfit to accurately evaluate the environment. We first propose an online novelty detection algorithm that operates on onboard perception system based features and also has anytime properties allowing it to deal reasonably with time-critical situations. We then extend this to the problem of change detection, a location-dependent version of novelty detection where the goal is to identify when areas of the environment have significantly changed from a previous traversal. This capability is well-suited for UGVs tasked with navigating a route repeatedly as in the case of supply routes or patrolling. Such systems would be the first safeguards for UGVs by identifying potentially dangerous situations during traversal such as the situation shown in Figure 3.2.

Finally, in Section 6 we discuss an online approach to candidate selection where a system must choose online between one of several operating modes and explore several applications of

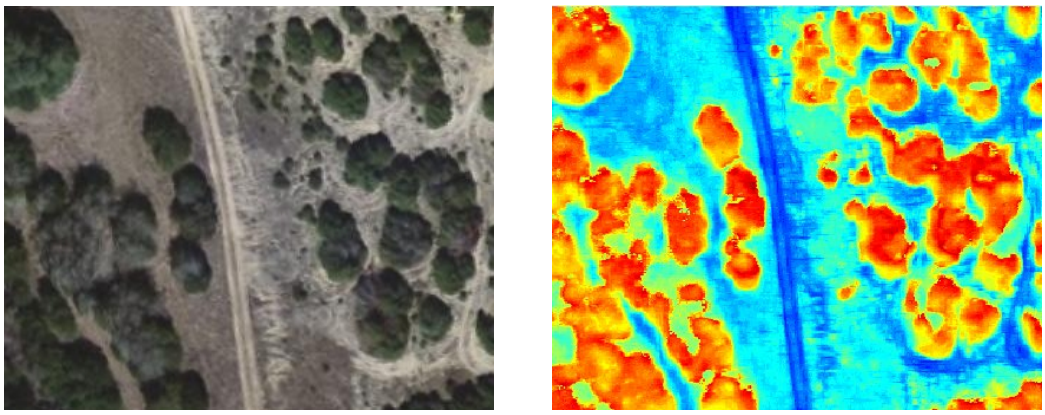


Figure 3.1: Sample results of online terrain traversal cost predictions from the Spinner UGV operating in Texas scrub lands using the system described in Section 4. 0.35 m resolution color overhead imagery used by our online learning algorithm is shown on the left and corresponding predictions of terrain traversal costs are shown on the right. Traversal costs are color-scaled for improved visibility. Blue and red correspond to lowest and highest traversal cost estimates, respectively.

such a system relevant to mobile robot navigation. We pursue this problem using an on-line, reinforcement learning approach. The system's goal is to learn to interpret available onboard and overhead sensor data in order to maximize its cumulative performance over the course of operation. This inevitably becomes a trade-off between exploring situations that will allow it to learn more about the world and exploiting those that appear to be optimal based on past experiences. In the case of learning to trade-off between autonomous and human tele-operation control, such a capability would enable a single operator to assist many UGVs, ensuring peak performance for the entire team with minimal human involvement.

We concede that the complexity and unpredictability of the real world forces robotic systems to have to adapt online. The key is to identify the feedback that allows online training, 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. Such techniques then 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 robotic systems.

3.2 Integration Into Autonomy System

Each of the approaches described above was primarily developed and evaluated independently of the others, allowing us to focus on rigorously evaluating the performance and benefits of each. This section will briefly discuss the expected use of each of these systems within an autonomy system on a fielded robot, as well as how all three could be used simultaneously for maximal benefit.

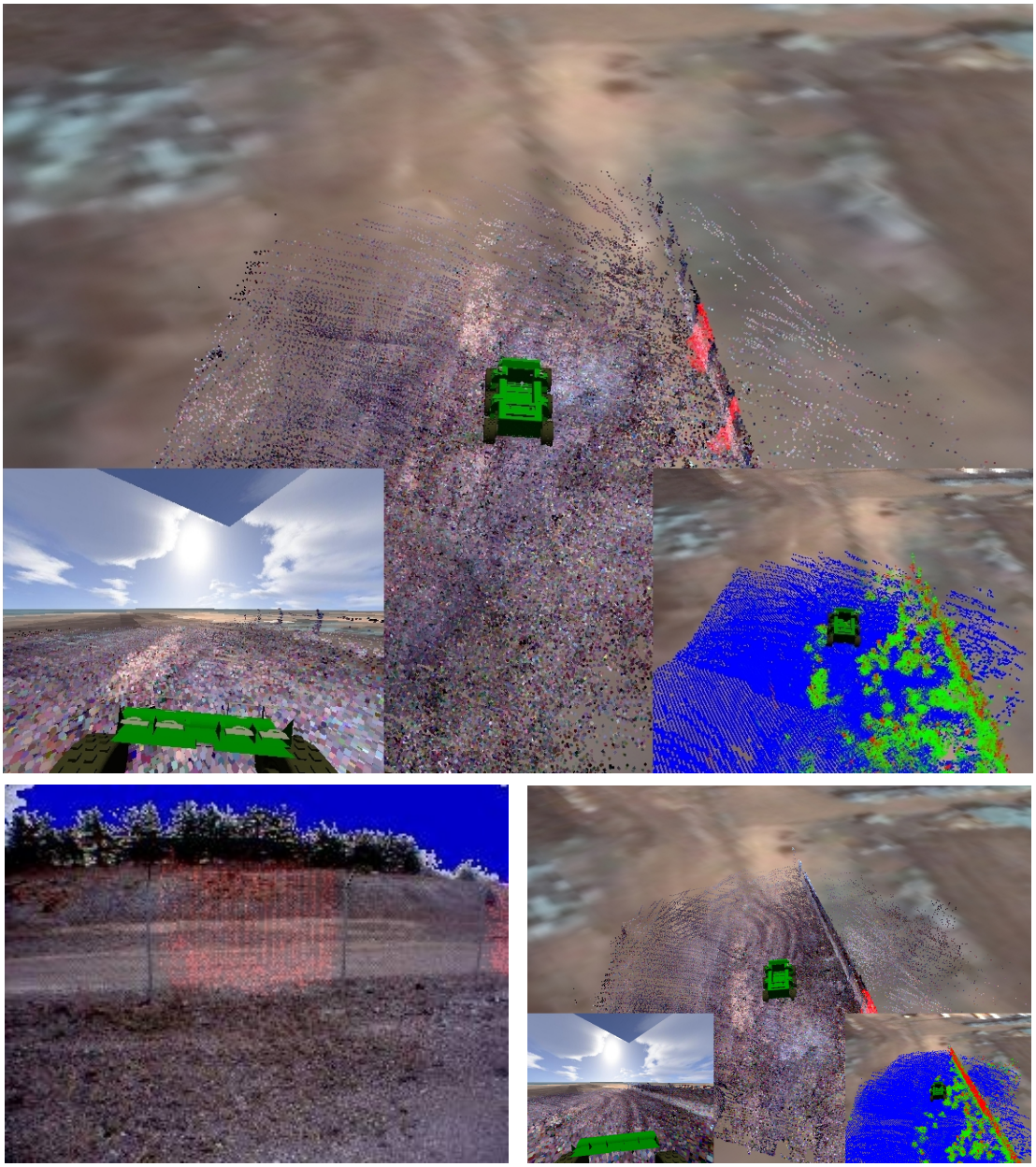


Figure 3.2: Sample result from online novelty detection algorithm onboard the Crusher UGV operating in western Pennsylvania. Chain-link fence was detected as novel (top and left, novelty shown in red) with respect to the large variety of terrain and vegetation previously encountered. After an initial stretch being identified as novel, subsequent portions of the fence are no longer flagged (right) due to the algorithm's online training ability. As with all future similar images, insets within the top image show a first-person view (left inset) and the classification of the environment by the perception system into road, vegetation, and solid obstacle in blue, green and red respectively (right inset).

3.2.1 Far-Range Perception (Chapter 4)

The purpose of the online-learning based far range perception systems described in Chapter 4 is to supplement the standard near-range perception system. In most cases the near-range perception system is assumed to be more accurate and robust due to its more powerful features (in fact that the far-range system uses the near-range system as an expert). An autonomy system should therefore fuse estimates from these multiple data sources using each sub-system's measure of confidence.

In locations where both near-range and far-range perception (most likely at close ranges) estimates are available, the near-range estimates should be favored. An exception to this may be in situations where the far-range perception system has some advantage over the near-range system, such as a better vantage point for negative obstacles as in the case of overhead data. In a majority of situations, however, the autonomy system would want to merge far-range perception estimates into the map used for planning, but overwrite them as more reliable near-range perception estimates become available.

3.2.2 Online Novelty and Change Detection (Chapter 5)

The novelty and change detection systems described in Chapter 5 allow a mobile robot to detect potentially hazardous situations before they result in a mission-ending mistake. In the case of novelty detection, the system should be initialized with a set of examples that captures the current perception system's experience base. Anything that is novel with respect to that model is assumed to be a potential hazard. Similarly, the change detection system requires perception features computed during a previous navigation of the intended environment, from which it will perform location-specific instances of novelty detection to detect changes.

A decision point occurs when a situation is detected as novel (or changed). The simplest thing to do is to assume the situation is hazardous and avoid it entirely (especially if a human tele-operator is unavailable or highly expensive). The other option is to request the aid of a human tele-operator to tell the robot how to deal with the situation. Deciding between these options could be handled by an intelligent uncertainty resolution approach as proposed in Section 7.2.3. Since a majority of situations could be simply avoided using such an approach, a much higher false-positive rate could be tolerated, allowing for improved overall performance.

Once a remote human operator has demonstrated to the system how to handle the situation in question several additional steps may be beneficial. If the human has specified that the situation in question is not dangerous (or can be adequately handled by the existing perception system), the novelty or change detection system can absorb this example into its model so that future similar situation are no longer flagged. On the other hand, if the remote operator needs to override the robot's default behavior, an approach such as that described in [112] can use the demonstrated path to encourage the robot's behavior to match that of the operator's, allowing the robot to handle similar scenarios in the future autonomously. In this way, a mobile robot can gradually adapt to unfamiliar domains, utilizing operator preference to improve future autonomous performance.

3.2.3 Online Candidate Selection (Chapter 6)

The online candidate selection system described in Chapter 6 allows the autonomy system to model and trade-off between multiple operating modes in real-time during operation. The choice points throughout navigation must be selected along the intended path, allowing the system to measure the performance of the chosen mode in between selections (we chose a distance-based discretization).

If the choices are among multiple autonomous modes then the frequency of mode changes have little impact so the algorithm can be used without further modification. If, however, one mode is far more limited or expensive than another (as in the case of a human tele-operator), additional care must be taken when utilizing the operating mode in question. We present a linear programming based approach to allow the system to optimize the use of a limited mode of operation, finding an optimal allocation based on the current model of each operating mode and the contextual information for the environment. As in the case of novelty detection, the intelligent uncertainty resolution technique described in Section 7.2.3 would allow the system to further improve its utilization of scarce resources by avoiding uncertain or difficult situations entirely.

3.2.4 Joint Integration

The three described approaches complement each other well and can be used in tandem within an autonomy system.

The far-range perception systems would operate at all times without human involvement, allowing the robot to better plan paths and react to upcoming obstacles.

The novelty (and change) detection and candidate selection systems are more tightly coupled, as they both potentially require limited human attention. As the perception system receives new data, the novelty detection system would first make a judgment on whether the new situation is novel. Only after the novelty detection system has finished processing the situation as described above would the candidate selection system be allowed to potentially query the human operator to improve performance.

Human attention is required for both handling novelty and changes in the scene as well as improving the performance of the system as directed by the candidate selection system. Because safety is of highest priority, the system needs to carefully regulate the use of this limited resource to ensure that the novelty and change detection systems will have access when necessary. Achieving this balance without significantly hurting performance is a difficult problem and another potential area of future work.

3.3 System Architecture

It is important to introduce the system specifications on the UGVs we operate with as we will be referring to many aspects of these systems throughout later chapters.


Figure 3.3: Spinner (left) and Crusher (right) robots used for experimentation throughout earlier thesis work. The natural terrain shown here is representative of the domains they operate within.

This work is focused on improving autonomous navigation in complex outdoor environments where a mission will often consist of a desired destination that must be reached in a reasonable amount of time, with the possible availability of varying quality overhead data. The lack of structure such as lane markings on a road or the flat drivable surfaces and binary traversability assumptions of indoor environments lays much of the complexity of this problem on the UGV's perception system. It must now be able to not only identify the presence or absence of obstacles, but also accurately interpret the relative risks to safety and progress of each situation. Bushes and small rocks may share similar qualities as seen by the perception system but differentiating between the two is vital to safe navigation.

The Spinner and Crusher UGVs of the UGCV-PerceptOR Integrated (UPI) Program that were used throughout the first half of this thesis (shown in Figure 3.3) are intended for operation in complex, outdoor environments, performing local sensing using a combination of ladar and camera sensors [124].

Due to the completion of the UPI Program in 2008, the remaining tests were performed using E-Gator vehicles of the R-CTA project (see Figure 3.4). These smaller UGVs were equipped with fewer sensors but the same core autonomy systems as that used on the Spinner and Crusher vehicles in order to be able to continue development and testing using similarly sophisticated perception features and sub-systems.

The autonomy system will be described briefly here with references to other literature provided for more details. Additionally, an article describing learning approaches utilized throughout the system as a whole can be found in [7].

3.3.1 Perception System

Figure 3.5 outlines the high level data flow within the autonomy systems of the UGVs used throughout this thesis. The perception system generates features from the color, position, density, and point cloud distributions of the environment [13, 70]. A large variety of engineered features that could be useful for this task are computed in real-time (see Figure 3.6) and the local

Figure 3.4: A modified John Deere E-Gator autonomous vehicle of the R-CTA program used for later portions of thesis work.

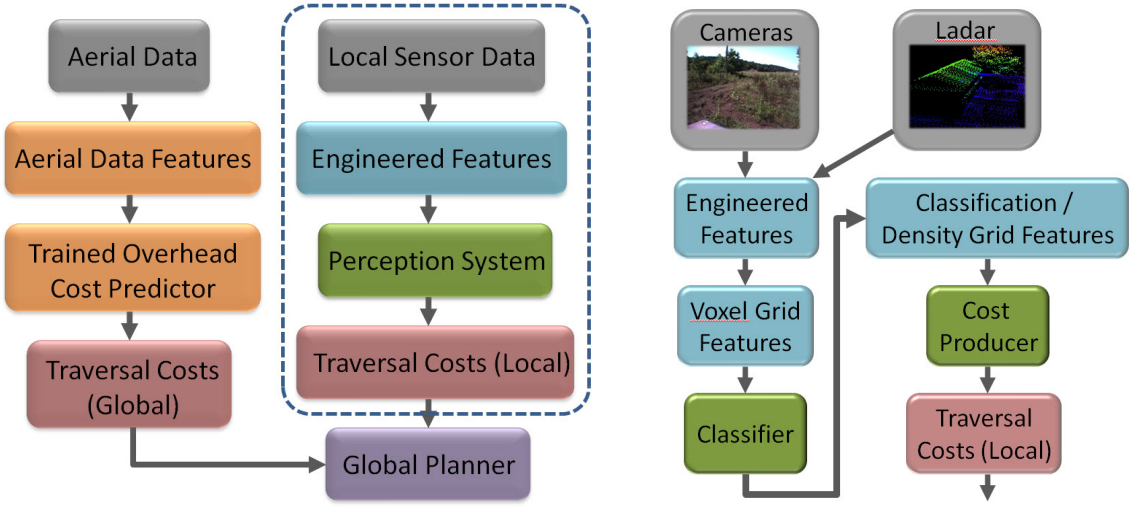


Figure 3.5: A high-level illustration of the perception system data flow (left) and a more detailed data flow of the perception system within the dotted box (right). Sensor data is interpreted to generate a variety of laser and camera based features. These features are grouped into 3D voxels which are interpreted by the perception system into traversal costs that the onboard planners can use for navigating to a goal.

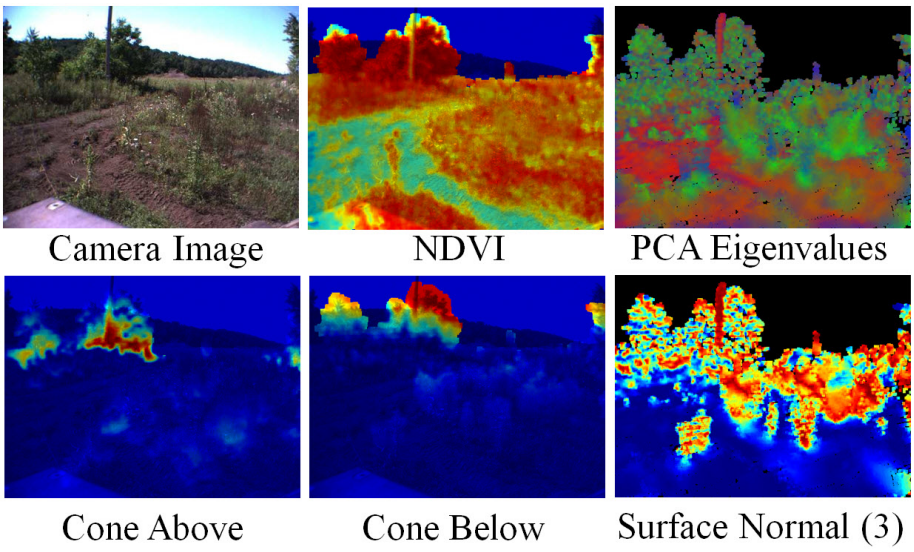


Figure 3.6: Example raw engineered features from the UGV's perception system. NDVI (normalized difference of vegetation index) is a useful metric for detecting vegetation [13]. PCA eigenvalue features are used to analyze the spacial distribution of the laser data. The cone features check whether there are any points in a cone starting from each point and are useful in identifying vegetation.

environment is segmented into columns of 20 cm^3 3D voxels (see Figure 3.7) in order to capture all potentially relevant information.

This vertical voxelization approach is effective for mobile robots since the presence of specific features at certain vertical positions is highly relevant to their impact on traversal cost. For example, solid objects at wheel height are likely to be small rocks while similar features higher off of the ground are more likely to be trees or man-made objects.

Sample voxel features computed for a possible environment are shown in Figure 3.8. Color and texture features for the environment are computed from camera imagery and tagged to each voxel by projecting the voxel into each image (colorized laser data can be see in Figure 3.8(b)). Consistency within color features is maintained using high dynamic range (HDR) imaging on pairs of images. The exposure times for the pairs of images are regulated using a controller that tries to minimize the number of pixels that are over-saturated or under-saturated between the images (see Figure 3.9 for an example of this approach).

Spatial features that are useful for describing the local shape of the point clouds can be seen in Figure 3.8(c). For example, the ground or a building wall will generate strong 'planar' features while bushes or tree canopy may have stronger 'spherical' features and poles or power lines will have stronger 'linear' features.

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, the traversal cost estimator uses an estimate of the terrain supporting surface to interpret voxel data. Voxels classified as ground-like which are above the estimated terrain surface could be dangerous rocks. Similarly, a tree limb at the level of the sensors can potentially disable the robot while at

Figure 3.7: Illustration of the perception system's voxelization of vertical columns within the environment and subsequent classification. The voxels here are actually much smaller within the system but are enlarged for demonstration purposes. In the perception system, each voxel is a 20 cm^3 cube. Due to the size of the Spinner and Crusher vehicles, 10 voxels in the vertical direction are computed at each location in order to include all potentially relevant information.

a higher elevations it may not interfere at all. Sample ground height estimates can be found in Figure 3.8(d). Further details on this system can be found in [143].

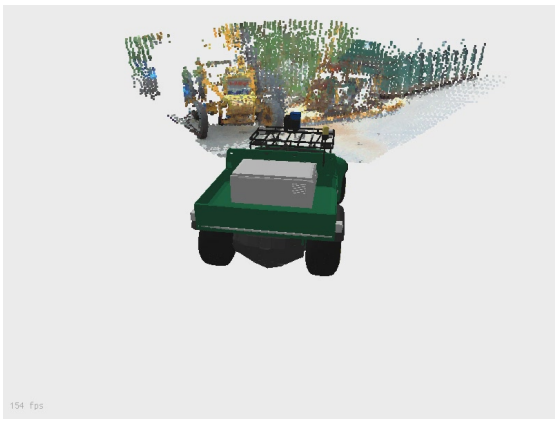
Each voxel, tagged with its corresponding features, creates a compact set of intermediate features for each location in the world that is more suitable for traversal cost computation. The system then interprets these features through hand-tuned or learned methods to create a final traversal cost for that location in the world that can be used for path planning purposes.

A useful and common abstraction for navigation is to represent the world as a 2-D horizontal grid around the robot. 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.

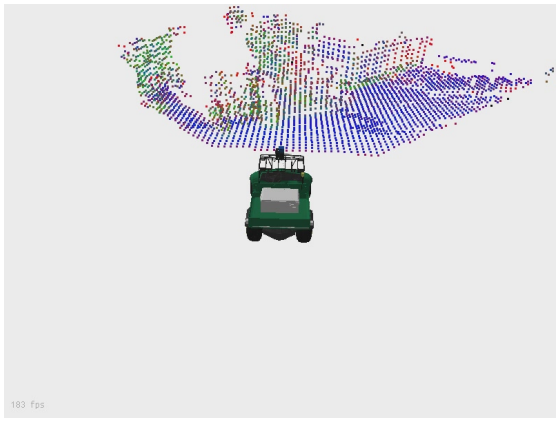
Traversal costs are interpreted as relative measures of mobility risk (our robot works with traversal costs in the range of 16 to 65535). For example, the robot's on-board perception system assigns traversal costs of 16 (the minimum) to roads while grass is assigned a traversal cost of 48, implying the robot would be willing to take a detour of three times the distance in order to stay on a road as opposed to driving over grass. Meanwhile, dense vegetation is often assigned traversal costs of over 10000 in order to encourage the robot to traverse elsewhere except under extreme necessity. The optimization of this cost function is an extremely difficult problem and rather than tuning this function by hand, a reinforcement learning approach described in [112] is used to quickly capture operator preferences.

The robot re-plans its global path in real-time by finding minimum cost paths through the environment using the Field D* algorithm [33] (an extension of the original D* algorithm [121]) and makes use of a sophisticated local planner to align its local behavior with the global plan [59].

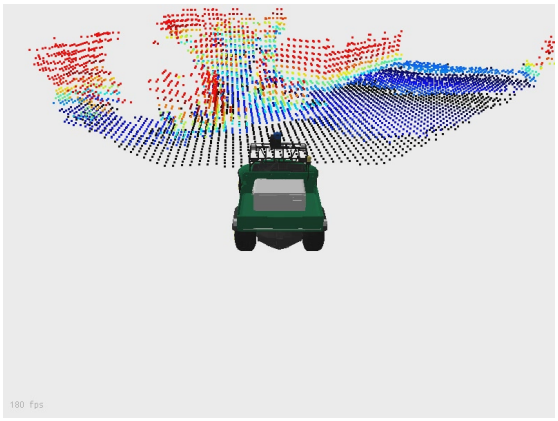
(a) Camera image of scene



(b) Colorized ladar data



(c) Spatial features: green, blue and red correspond to spherical, planar and linear features respectively



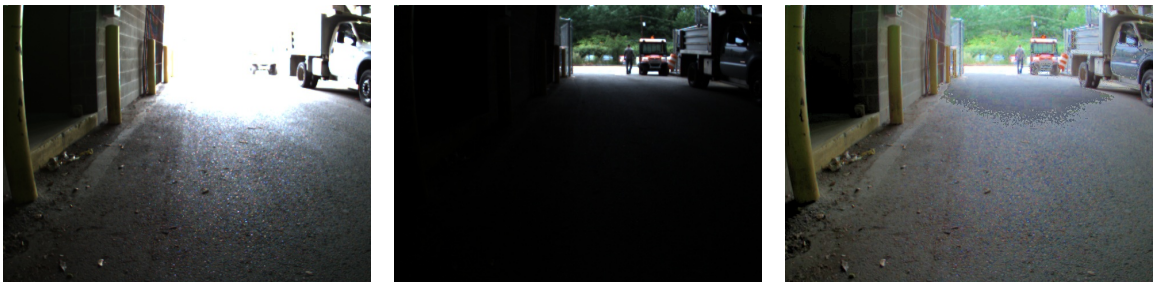
(d) Computed height off of ground plane

Figure 3.8: Example output of aspects of the perception system for shown scene.

An extensive log playback and processing infrastructure allowed us to develop and optimize many of our approaches offline.

3.3.2 Overhead Data Usage

Improvements in both path safety and efficiency were achieved on the UPI program by taking advantage of prior overhead data when available. Prior data can come from a variety of sources. Low resolution imagery (1 meter resolution gray scale) and topographic data (10 to 30 meter resolution) are already available for most of the world. In addition, higher resolution imagery and dense three-dimensional (3-D) data can be collected commercially upon request. Traversal cost maps for the environment can be produced from this data a priori using a hand-trained system or human demonstration for aiding online global path planning. The robot's onboard perception system is then used during navigation to fine-tune prior traversal estimates and adjust for areas of limited aerial visibility or changes in the environment from the time data was gathered.



(a) High exposure (b) Low exposure (c) Combined image

Figure 3.9: High-Dynamic-Range imaging is used within the perception system to maintain consistency in image-based features. Pairs of images with different exposure times are used to maximize the number of pixels in the image that are adequately and consistently saturated.

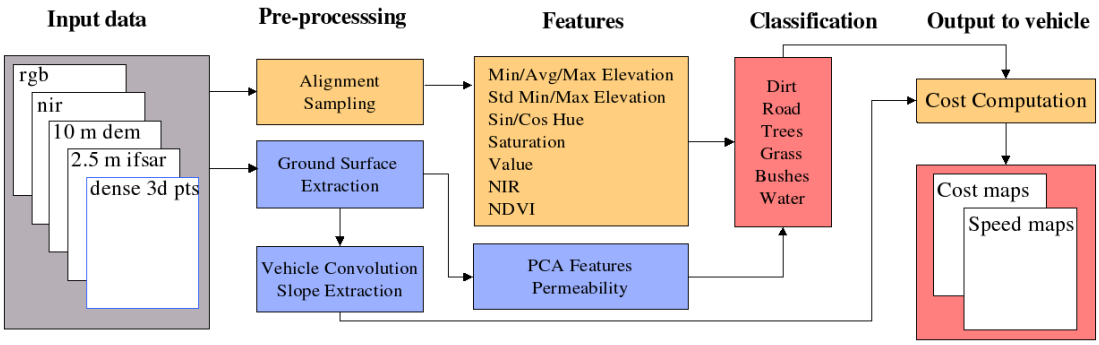


Figure 3.10: Overview of overhead data processing system: from raw data to cost maps.

In the human-supervised overhead cost prediction system, traversal costs are computed from a combination of semantic and geometric data as described in Figure 3.10 [110]. Semantic information of the terrain is obtained through supervised classification using features extracted from imagery and 3-D data [114]. A neural network with one input node for each feature, one hidden layer, and one output node for each desired classification category is used. Each terrain class is assigned a traversal cost by a human operator designed to mimic the behavior of the onboard perception system. Mobility analysis is then performed using the ground surface recovered from 3-D data (see Figure 3.11) or an available elevation map and traversal costs are assigned to reflect the capabilities of the vehicle based on computed parameters such as roll, pitch and ground clearance. Traversal costs are independently computed using the results of terrain classification and vehicle mobility analysis for each location in the world and summed to produce final traversal costs as shown in Figure 3.12.

Another approach used for taking advantage of overhead data sources is to have the system learn from demonstrated behavior [111]. Once provided with examples of how a domain expert would navigate based on the data, an imitation learning approach can learn mappings from raw data to cost that reproduce similar behavior. Because it is often difficult to pick and hand-tune traversal costs to achieve acceptable behavior, this approach produces cost functions with less human interaction and often better performance.

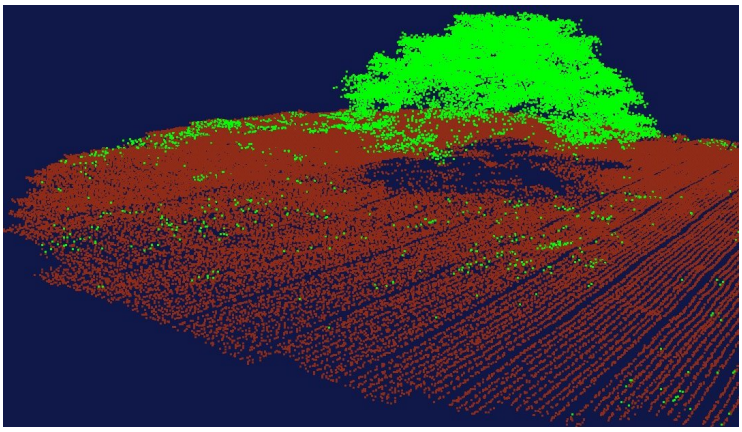


Figure 3.11: 3-D points are classified as ground (brown) or non-ground (green).

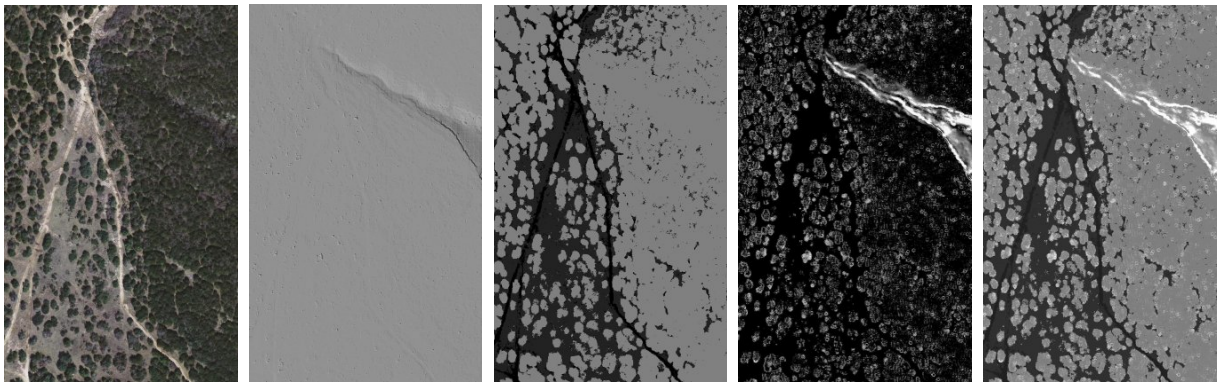


Figure 3.12: Sample results of cost map production for generally flat Texas terrain with a large vegetation variety. From left to right: an image of the environment, extracted ground surface, classification cost map, mobility cost map and final traversal cost map (sum of classification and mobility based cost maps). Each image is 368×595 meters.

While these techniques for interpreting overhead data have been shown to significantly improve navigational performance, the requirement of manual retraining each time a new environment is encountered is a notable limitation that we address in this thesis.

Chapter 4

Onboard and Overhead Robot Perception in Unfamiliar Domains

Autonomous robot navigation in unstructured natural environments has been demonstrated extensively in a large variety of terrain, sensor payload and mission scenarios. Even though powerful at sensing, modeling, and interpreting the environment, these systems required significant tuning of parameters, either by hand or supervised training, to best adjust their algorithms to the local environment where the tests are conducted.

This highlights a common problem that arises in mobile robotics where potentially powerful sensor data and features are often difficult to take advantage of because they are situation or location specific. As mentioned previously, outdoor robot navigation can benefit from the now widespread availability of high quality overhead imagery and elevation data from satellite and aircraft. With this overhead data, many of the difficulties associated with autonomous robot operation can be alleviated, even with the coarsest of terrain resolution. Systems can then dispense with myopic exploration and instead pursue routes that are likely to be effective. Unfortunately, features computed from such sources can vary greatly due to diversity in terrain, environmental conditions and sensor varieties. This often invalidates pre-trained systems so the necessity for frequent manual retraining reduces the appeal of such approaches.

In much the same way, the complete use of onboard sensor data (such as ladar), if interpreted correctly, can help a robot make better decisions and increase traversal speed through improved knowledge of the environment. Unfortunately, as the complexity of environments increases, a robot's perception system must be engineered to evaluate its environment by first computing a set of intermediate features such as ground slope, object density, and vegetation classification. While such features are consistent and generalizable, fixed techniques that successfully interpret onboard sensor data across many environments begin to fail past short ranges as the density and accuracy necessary for such computation quickly degrade and features that are able to be computed from distant data are very domain-specific (see Figure 4.1). This limits the effectiveness of such perception systems to a proximity of about 15 meters even though sensor data is often available at much higher ranges.

Building systems that can reliably interpret these scope-limited features is far from easy as even the smallest variations in lighting, season, terrain or even sensor calibration can have sig-

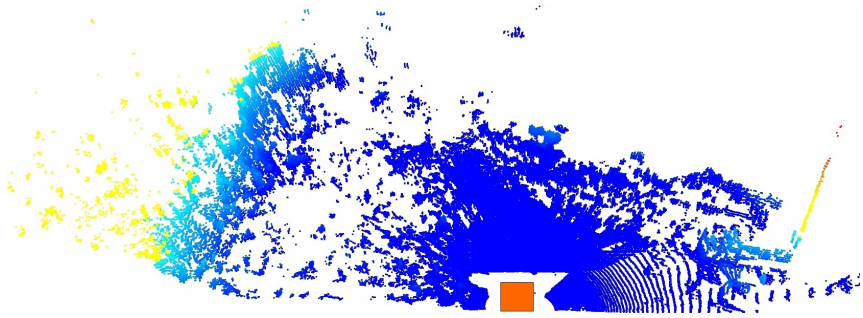


Figure 4.1: Typical ladar response from vehicle's perception system. Ladar points are color coded by elevation with lowest points appearing in blue and highest points appearing in yellow. Vehicle position is shown by the orange square. Notice the large drop in ladar response density (especially on the ground) as distance from the vehicle increases. Large objects such as the trees on the left generate ladar responses even at far ranges but are difficult to interpret through fixed techniques across different environments.

nificant effects on data. Additionally, such estimates must be well calibrated with other onboard perception estimates of the terrain, or system performance may suffer. Developing approaches that can leverage these potentially powerful resources on an autonomous robot can significantly improve the versatility of many unmanned ground vehicles by allowing them to traverse highly varied terrains with increased performance.

One way to address such limitation is through on-line self-supervised learning where the autonomous system adjusts itself via perception and interaction with the environment. We propose to address the problem of learning and inferring between two heterogeneous data sources that vary in density, accuracy and scope of influence. The objective is to generalize from one data source, viewed as a reliable estimate, to be able to work with another, which may be high performance (e.g., long range or high accuracy) but difficult to generalize to new environments. We frame the problem as a simple, linear probabilistic model for which inference results in a self-supervised online learning algorithm that fuses the estimates from the two data sources. We also explore the advantages of this framework including reversible learning, feature selection, data alignment capabilities, reliable use of multiple estimates, as well as confidence-rated predictions [115].

4.1 Related Work

When navigating in an environment without full knowledge, robotic systems primarily rely on on-board perception systems. There are cases, however, when it is not possible to get an adequate understanding of the environment from the vehicle-based view of terrain without sacrificing speed or path optimality. For example, a vehicle navigating at high speeds in off-road environments may be unable to react to negative obstacles such as large holes and cliffs without the prior availability of overhead data. Even when the vehicle can safely navigate an environment, aerial sensing can dramatically improve path planning performance by detecting large obstacles such as buildings, forests and bodies of water as well as areas of preferable terrain such as roads.

Even low resolution prior data can provide significant improvement to vehicle performance,

as demonstrated by numerous simulations in [134]. In [132], low resolution (25-30 meter) elevation maps were used to aid long distance planning. [105] demonstrated the extraction of features (roads, trees, water, etc.) from aerial surveys, for later correlation by an autonomous vehicle.

Similar research conducted by Charaniya et al. classified terrain into roads, grass, buildings, and trees using aerial LiDAR height data, height texture, and signal reflectance, achieving classification rates in the range of 66%-84% [21]. Cao et al. attempted to identify man-made objects from overhead imagery [16]. Knudsen and Nielson attempted to classify buildings using a previously available GIS database and RGB information for an environment [62].

The DARPA PerceptOR program contained an important prior data component. Aerial LiDAR data was used to predict vehicle roll and pitch over stretches of terrain, as well as to detect vegetation [123, 137]. This information was used to generate prior cost maps for use in global planning.

Within the UPI Program, several overhead data processing techniques were used extensively to achieve safe and efficient navigation over many kilometers (see Section 3.3) [110, 112, 114].

While such overhead data interpretation techniques proved crucial to successful navigation in some difficult environments, they were limited by the fact that they required human training or demonstration prior to use in new domains. Also, as discussed earlier, training onboard perception systems for general far-range use is infeasible due to the large variability in the features generated past short distances from the robot. As a result, such systems are often engineered or trained to perceive the environment in close proximity to the robot where features more often generalize well across many domains. In such cases the large portion of sensor data that cannot be interpreted through such techniques is often discarded.

Our approaches is one of several to use self-supervised learning to deal with the common robotics situation where data sources that, while highly relevant, are domain specific. For instance, camera imagery can potentially detect unpaved road in the desert significantly farther than some ladar-based systems can. Unfortunately, such data can prove very resistant to automated interpretation. In particular, classifiers that prove to be powerful indicators of road in a particular area often do not generalize to new conditions. Detecting such roads from a distance in a self-supervised manner proved to be a crucial component in Stanford Racings winning Grand Challenge entry [25]. A similar version of this approach using reverse optical flow has also been proposed [73] and subsequently extended to off-road navigation [75].

Similarly, [119] presents a self-supervised approach for estimating terrain roughness from laser range data. Sensor, terrain and vehicle varieties contribute large errors that must be considered when making estimates. This system learns to model error by providing its own labels of terrain roughness in real-time from actual shock measured when driving over the target terrain. The vehicle in effect capitalizes on its ability to measure terrain roughness from the vehicle's inertial sensors to its range sensors.

Others have built systems that optimize parameters online using real-time performance as feedback. For example, such a technique allowed an adaptive motion planning system on an autonomous excavator to learn to perform at levels approaching skilled operators [98].

Numerous other research efforts have taken advantages of such techniques for obstacle avoidance using monocular camera [72, 138], estimating the depth from monocular imagery [84], au-

tomated learning of noise parameters in Kalman filters [1], terrain traversability classification [60], slip prediction [2], and estimating ground height from an autonomous tractor [143].

4.2 Approach

4.2.1 Formalization

We approach the problem of leveraging the powerful, but difficult to generalize, features in a Bayesian probabilistic framework using the notion of scoped learning [11]. The scoped learning model admits the idea of two types of features: "global" and "local". Global features are generally useful, and their predictive power extends well to new domains, while local ones, which, although often very powerful, typically generalize poorly and are more difficult to take advantage of in a consistent way. These local features have *scope* that is limited to one particular domain. We wish to apply our system to extend the scope of such features to many possible domains. For our canonical problem of learning to leverage the extended range of overhead and far-range sensor data, these names may prove counter-intuitive, so we refer to them instead as *general* and *locale-specific* features. From this point on, "global" and "local" will refer to the proximity to the robot. Features generated from dense, vehicle-based ladar perception serve as our general features, while features generated from overhead based imagery and elevation data and far-range sensor data serve as our locale-specific features. The latter are particularly valuable to mobile robots because of their extended range and widespread availability.

Model. The scoped learning approach is a simple probabilistic model (shown graphically in Figure 4.2) that captures this notion of features that have scope. The outer plate L represents in graphical model notation that there are independent locales in which the model will be applied [57]. These correspond to new areas of the world in which our robot will operate.

Within the plate, we see a sequence of locale-specific features and corresponding general feature-based estimates. At each point in the sequence, we wish to make predictions about c (either all or a subset of them.) Here, c is the true variable we wish to predict, and $\tilde{c}$ is an estimate of that variable coming from the general features, while $\mathbf{x}$ are our locale-specific features. The parameters β common to the locale (plate) capture the relationship between locale-specific features and the variables of interest c. The length of our sequence is n.

This learning model captures the idea of self-supervised learning [58] in a Bayesian framework and extends the idea to integrate both the general feature-based estimates and the self-supervised locale-specific estimates. Driven by our application, we are particularly interested in the online *regression* case[1] where the goal is to learn to infer the true continuous values c_i in an online fashion as general feature-based estimates $\tilde{c}_i$ become available. We choose a simple model for c as a function of the k locale-specific features $\mathbf{x} = (x^1, \ldots, x^k)$ by modeling the distribution for c given $\mathbf{x}$ as a Gaussian with mean a linear function of $\mathbf{x}$ and with a variance of σ_l^2, giving us

[1]The original scope learning work [11] was developed in the context of classification using discrete features, generative descriptions of those features, and in batch.

the following expected value for c:

$$E(c|\beta, \mathbf{x}) = \beta^T \mathbf{x} \tag{4.1}$$

We assume that the estimates from the general feature-based predictors have Gaussian noise and thus are distributed:

$$\tilde{c} \sim \text{Normal}(c, \sigma_g^2)$$

We take the σ_g and σ_l to be hyper-parameters lying outside the locale-specific plate.

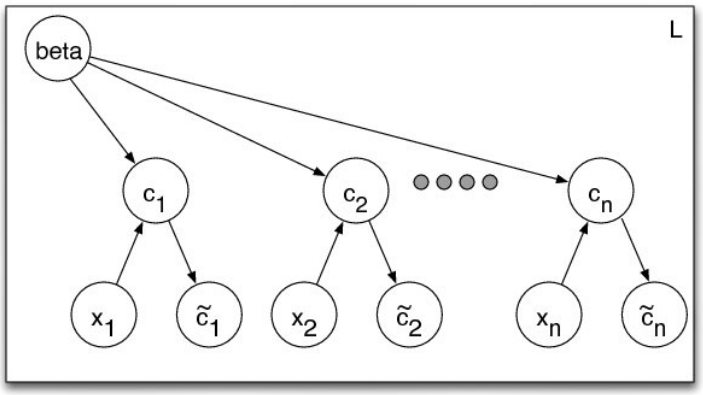


Figure 4.2: Graphical depiction of the scoped learning model. c_i are the true traversal costs for the locations in the world and $\tilde{c}_i$ are the estimates for those locations from the perception system. x_i are the locale specific features that we are trying to infer from where β are the model parameters that govern the relationships between x_i and c_i. Hyper-parameters, including priors on the locale-specific parameters β and noise variances, lie outside the plate indexed by L and are not depicted.

To clarify this model using a real-world example, consider the case of learning online to predict traversal costs from overhead imagery data. In this scenario, c_i are the true traversal costs for locations in the world and $\tilde{c}_i$ are the general feature-based estimate for traversal costs of those location coming from the UGVs onboard perception system. These estimates are gathered online and are only available for a small subset of the locations in the world. The locale-specific features $\mathbf{x}_i$ are computed from overhead imagery for the environment and include color and texture based features as discussed later. In the case of overhead imagery, features $\mathbf{x}_{1\ldots n}$ are available for all location, but we do not have a way to interpret these features a priori. We assume that c_i is a linear function of $\mathbf{x}_i$ governed by some weights β. As we traverse through the environment and observe additional perception system-based estimates $\tilde{c}_i$, we can refine online our best estimate of the mapping from overhead imagery-based features $\mathbf{x}$ to c.

Inference. We develop the inference for the model in an online fashion. Given a new data point $\tilde{c}_i$ estimating the true variable c_i, our goal is to compute new estimates[2] of the variables c_j, assuming we have already seen data $D = \{\mathbf{x}_{1\ldots n}, \tilde{c}_{1\ldots i-1}\}$. We can compute this by integrating over the uncertain parameters β which describe the relationship between the true variable and the local features.

[2]We assume that our prior on β is a priori independent of the features $\mathbf{x}$ so that inference will remain the same even in the case where the features become available in some sequence.

$$p(c_j|\tilde{c}_i, \mathbf{x}_i, D) = \int d\beta \; p(c_j|\beta, \tilde{c}_i, \mathbf{x}_i) p(\beta|\tilde{c}_i, \mathbf{x}_i, D)$$

We can compute the required distribution over β as:

$$p(\beta|\tilde{c}_i, \mathbf{x}_i, D) \propto p(\beta|D) \int dc_i \; p(\tilde{c}_i|c_i) p(c_i|\beta, \mathbf{x}_i)$$

In our linear-Gaussian model, this can be understood as revising the posterior distribution from $p(\beta|D)$ in light of a Gaussian likelihood that takes into account noise from both general and locale-specific features.

Our computation of the posterior distribution $p(\beta|\tilde{c}_i, \mathbf{x}_i, D)$ is as follows. We first initialize our distribution to the prior distribution $p(\beta)$. Then, for every training example i, we multiply our distribution by $p(\tilde{c}_i|\beta, \mathbf{x}_i)$. Since the prior distribution and $p(\tilde{c}_i|\beta, \mathbf{x}_i)$ are normal, the posterior distribution is also normal. We use the notation $\hat{\beta}$ to represent the mean of the posterior distribution and V_β to represent the variance. Thus, computing $p(\beta|\tilde{c}_i, \mathbf{x}_i, D)$ is performing a self-supervised learning using a Bayesian linear regression model with noise variance $\sigma_l^2 + \sigma_g^2$.

We use our current estimate of the posterior distribution when we want to predict a future outcome c_j. We are interested in predictions in two cases: first, when we have no general feature-based estimate $\tilde{c}_j$ for a particular c_j, and second, when such an estimate is available. In the first case, the predictive distribution $p(c)$ has mean $c_p = \mathbf{x}^T \hat{\beta}$ and variance $\sigma_p^2 = \sigma_l^2 + \mathbf{x}^T V_\beta \mathbf{x}$ [37]. When we also have an estimate $\tilde{c}_j$, inference combines these two estimates:

$$p(c_j) = \text{Normal}(\sigma_p'^2 (\frac{c_p}{\sigma_p^2} + \frac{\tilde{c}_j}{\sigma_g^2}), \sigma_p'^2)$$

where

$$\sigma_p'^2 = \frac{1}{\frac{1}{\sigma_p^2} + \frac{1}{\sigma_g^2}}.$$

We note that it is possible to compute the posterior distribution in batch, but we prefer to maintain an estimate of the posterior distribution as we receive general feature-based cost estimates so that we may immediately apply our algorithm to new data.

4.2.2 Advantages of the Bayesian Learning Approach

Using the online Bayesian scope learning model provides a number of important benefits.

Confidence Rated Prediction. The variance estimate provided by our algorithm for the probability of each c can be used as a metric of confidence in the prediction. If a situation arises in which we must choose which one of several predicted outcomes to trust, we could simply use the one with the smallest variance.

Learning of the Hyper-Prior and Feature Selection. Our algorithm depends on a number of hyper-parameter terms that may be chosen based on data from multiple locales. We discuss ways to choose the noise variance terms σ_l and σ_g in section 4.2.1 and the prior distribution

on parameters β in section 4.3.3. The prior distribution $p(\beta)$ is an isotropic Gaussian that is independent for each weight, and each weight is dependent on a shared hyperparameter α that moderates the strength of our belief over the values our weights β might take. That one α value controls the inverse variance of each weight β. We can modify our prior $p(\beta|\alpha)$ to consist of K hyperparameters, with each α_k independently controlling the inverse variance of each weight and define hyperpriors over all the α_k values. We can then use Tipping's hyperparameter re-estimation algorithm to do feature selection (see Algorithm 1) [130]. In this way, we can both automate feature selection and bias our algorithm to prefer certain features for new locales [93].

Algorithm 1 Hyper-parameter re-estimation procedure

1: **given:** Initial values for all α_k and σ_l^2
2: **while** α has not converged **do**
3: Compute the mean $\hat{\beta}$ and covariance V_β of the distribution of our weights β
4: For all K features, $\gamma_k \leftarrow 1 - \alpha_k V_{\beta kk}$
5: For all K features, $\alpha_k \leftarrow \frac{\gamma_k}{\hat{\beta}_i^2}$
6: **end while**
7: **return** α

Once our α_k values have converged, we can remove any feature x_k with a corresponding optimal α_k value that tends toward ∞. Since an α_k value controls the inverse variance of each weight β_k, an α_k value that tends toward ∞ implies that the mean of the weight β_k value tends toward 0. Thus, we can remove that feature x_k since its weight β_k value of 0 would remove its contribution to predicting the output $\tilde{c}$.

Reversible Learning. A problem that often arises in online learning is the handling of multiple estimates of a particular quantity. For instance, in our canonical example, our general feature-based estimates $\tilde{c}_i$ may improve as we get closer and denser laser readings of the terrain. It is not appropriate to treat these as independent training examples: while they may differ in their variance, they are generally highly correlated. Neither is it useful to simply take the first estimate available: often this is a poor substitute for all the data.

In our model, we maintain an exact posterior distribution that lies inside the exponential family and assume that the observation errors are independent and have equal variance. We may therefore effectively *remove* the effects of a previous training example j on our posterior distribution by dividing out $P(\tilde{c}_j|\beta, \mathbf{x}_j)$, the likelihood term we had used to include it in the posterior [37]. In this way, we always have an estimate of the posterior distribution of β using the current best estimate $\tilde{c}_i$. Minka has developed an alternate use of this "removal trick" for approximate inference [85].

An extended discussion of the Bayesian Linear Regression algorithm, its derivation and full use can be found in Appendix A.

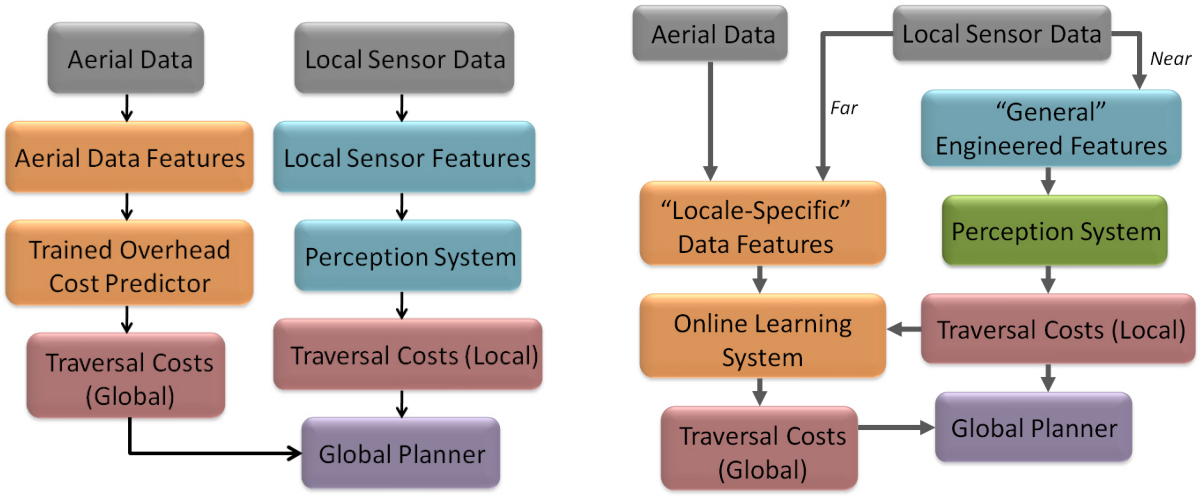


Figure 4.3: In the baseline system (left), aerial data can only be used after the cost prediction module is trained either through human labeling or demonstration. Utilizing the online learning framework (right) allows the perception system to learn mappings from locale-specific data features (overhead or far-range sensor data) to locally computed terrain traversal costs (computed from general features) and make prediction elsewhere in the environment.

4.3 Application to Mobile Robotics

We demonstrate this approach in the context of long range navigation onboard the Spinner and Crusher UGVs described in Section 3.3 during various field tests in complex natural environments. These environments in western Pennsylvania and Texas consisted of a variety of vegetation ranging from grass and short bushes to large, dense forests as well as other diverse obstacles such as large rocks, hills, dunes and ditches.

The information flow within the baseline system discussed in Section 3.3 is modified as shown in Figure 4.3. As the UGV traverses an environment, it utilizes its on-board perception system and these difficult to interpret features (computed from overhead imagery and elevation data or far-range sensor data) to learn the mapping from these features to computed terrain traversal costs in order to predict traversal costs elsewhere in the environment where only overhead data or far range sensor data is available, effectively extending the range of the vehicle's local perception system and allowing more effective navigation of the environment.

4.3.1 Terrain Traversal Cost Prediction

We chose to predict traversal cost rather than intermediate results such as slope, density, or presence of vegetation because traversal cost is the metric that most closely governs a vehicles navigation strategy through an environment. Our robots perception system is proficient at effectively assessing terrain traversal costs, so it is desirable to be able to mimic its predictive abilities. We therefore use estimates from the robots perception system to evaluate the accuracy of traversal cost predictions.

Overhead Data Features. A set of feature maps for the vehicle's environment was generated

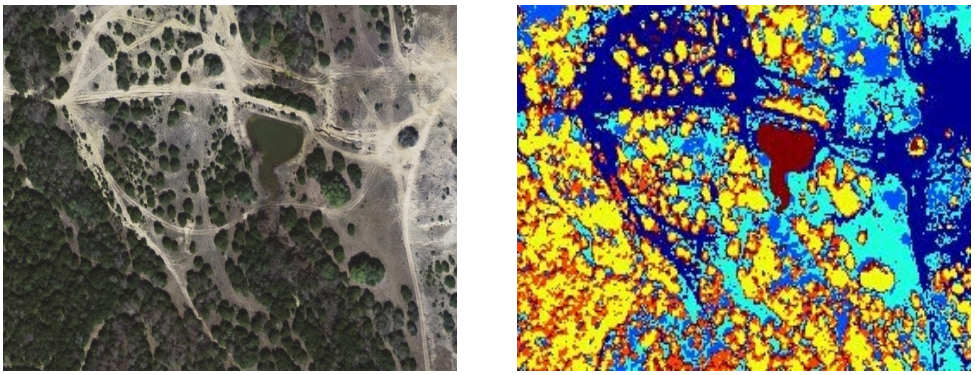


Figure 4.4: Sample clustering results from using the Gaussian Mixture Model algorithm on generated features. Overhead color imagery data used to generate features (left) and resulting clustering into six clusters (right). Membership features were generated by computing the fractional degree of membership of each pixel in each cluster.

from each overhead data source for use as inputs to the algorithm (these are our *locale-specific* features as defined in Section 4.2.1). In our implementation, HSV (hue, saturation, value) features were used to represent color imagery data while the pixel intensity of the black and white imagery data was used as a single feature. Raw RGB (red, green, blue) color data was inadequate for our approach due to its sensitivity to illumination variations. A hue (in degrees) of α was represented by the pair of values, $sin(\alpha)$ and $cos(\alpha)$ to address the continuity problem at hue values close to $0°$.

A feature containing the maximum response to a set of ten Gabor filters at various orientations centered at each pixel was also generated to capture texture in each type of imagery. Additional features for the black and white imagery data were generated by computing the means and standard deviations in intensity within windows of five meters around each pixel. Additional elevation-based features were computed as described in Section 3.3.2 and [110] when such data was available. All features were rescaled to the $[-1, 1]$ range and a constant feature was also included.

Finally, clustering of all previously computed features was performed that helped the algorithm to overcome the limitations of a linear model and easier identify patterns in the feature input space. Clustering of the input data was done through Gaussian Mixture Models in order to generate membership features by assigning each data point a fractional degree of membership in each output cluster (see Figure 4.4) [29]. Six clusters were chosen for our implementation.

The characteristics of an environment change with varying conditions. However, even outdated overhead data can be useful since most distinct areas in an environment will maintain uniformity in their characteristics despite these variations. By relaxing restrictions on the recency of overhead data, our algorithm further increases its impact on improving robot navigation.

Far-Range Sensor Data Features. Ladar and camera data were used to create the far-range sensor features used for training (once again, these are our *locale-specific* features as defined in Section 4.2.1). Ladar points in the environment were tagged with color values from the camera sensors by computing the pixel the ladar would appear in within the camera image. Color features were computed just as with overhead data. Additionally, the positions of the ladar points were used to compute the maximum vertical point spread and standard deviation of point heights. In

order to incorporate contextual information about the neighbors of a location, similar features were computed from the location's neighbors within a small window. A constant feature was included as well. As a robot travels toward a location, features for that location are computed regularly (to account for variations in features due to distance from the robot) and stored to be used as possible future training examples.

Finally, it should be noted that the *lack* of ladar data in an area in front of the robot can serve as a confirmation of free space since ladar hits are rarely available on road or grass past about 30 meters due to the angle with respect to the sensor. We identify such free space by tracing away from the robot until a sensed object is encountered (up to 40 m away) and setting a tracing feature in all encountered cells. This features is the sole feature for such areas.

In some applications, subsets of features may not always be available. For example, in the case of far-range sensor data, some laser data may fall outside of the field of view of the onboard cameras making color-based features unavailable for those locations. In such a scenario, an independent instance of the learner can be trained simultaneously for each potentially possible set of features. New examples can then be sent to the learner that is trained to handle its available feature set.

Such techniques may be used to generate features from any combination of data sources gathered through a variety of methods.

4.3.2 Training and Prediction

Because traversal costs act as distance ratios as described in Section 3.3.1, errors in traversal cost estimates in low-cost areas are more detrimental than similar errors in high-cost areas. An error of 100 to an area of extremely high traversal cost would have negligible effect, while the same error at an area of desirable terrain would radically change the behavior of the robot.

In order to work with the linear model used by our algorithm, we deal with traversal costs within our algorithm on a logarithmic scale, converting from the normal traversal cost space for the purposes of training and prediction. The Gaussian error assumption embedded in our probabilistic model is a much better approximation when we measure error on this scale. Unlike in the regular traversal cost space, small errors in the log space lead to small errors in the traversal distance ratios.

Training examples are constructed from $\mathbf{x}_i$, the vector of feature values from either overhead data or far-range sensor data, and $\tilde{c}_i$, the average of all traversal cost estimates that have been calculated within the corresponding area. As with many robotic systems, the performance of our robot's on-board perception system quickly degrades as the distance from the robot increases (due to the lowered accuracy and density of sensor data), so the quality of a training example is measured by its proximity to the robot. Rather than struggling to decide at which point to utilize an example for training, the reversible learning capabilities of our algorithm allow us to maintain an optimal level of predictive abilities by ensuring that only the highest quality data available impact its state. As the robot approaches locations that had previously been used for training, obsolete examples are *unlearned* in favor of higher quality training examples available

for those areas. Estimates greater than 12 m from the robot are ignored since such estimates are very unreliable and would only corrupt the quality of training in cases where they cannot be replaced with better estimates.

An example of this training process can be seen in Figure 4.5. As the vehicle explores more of the environment, the greater sample of training data allows it to more accurately interpret the locale-specific data sources. Notice how the shadow from the tree at the top right is initially estimated incorrectly as a very high-cost area (Figure 4.5c) but as the robot explores more of the environment, it begins to recognize its error and lower its estimate (Figure 4.5d).

As the algorithm acquires more training data, its predictive performance improves, allowing it to revise previously made traversal cost estimates. The algorithm specifies a degree of confidence for each prediction based on the similarity of the example to past training data (as indicated by the variance estimate), so predictions in which the algorithm lacks confidence can be ignored in favor of an alternative source of predictions or a default value. Notice how in Figure 4.5b the algorithm is able to identify its estimates for the trees in the environment as areas of low confidence (shown in blue) until the robot first encounters the tree below its starting position and is able to refine estimates for similar areas.

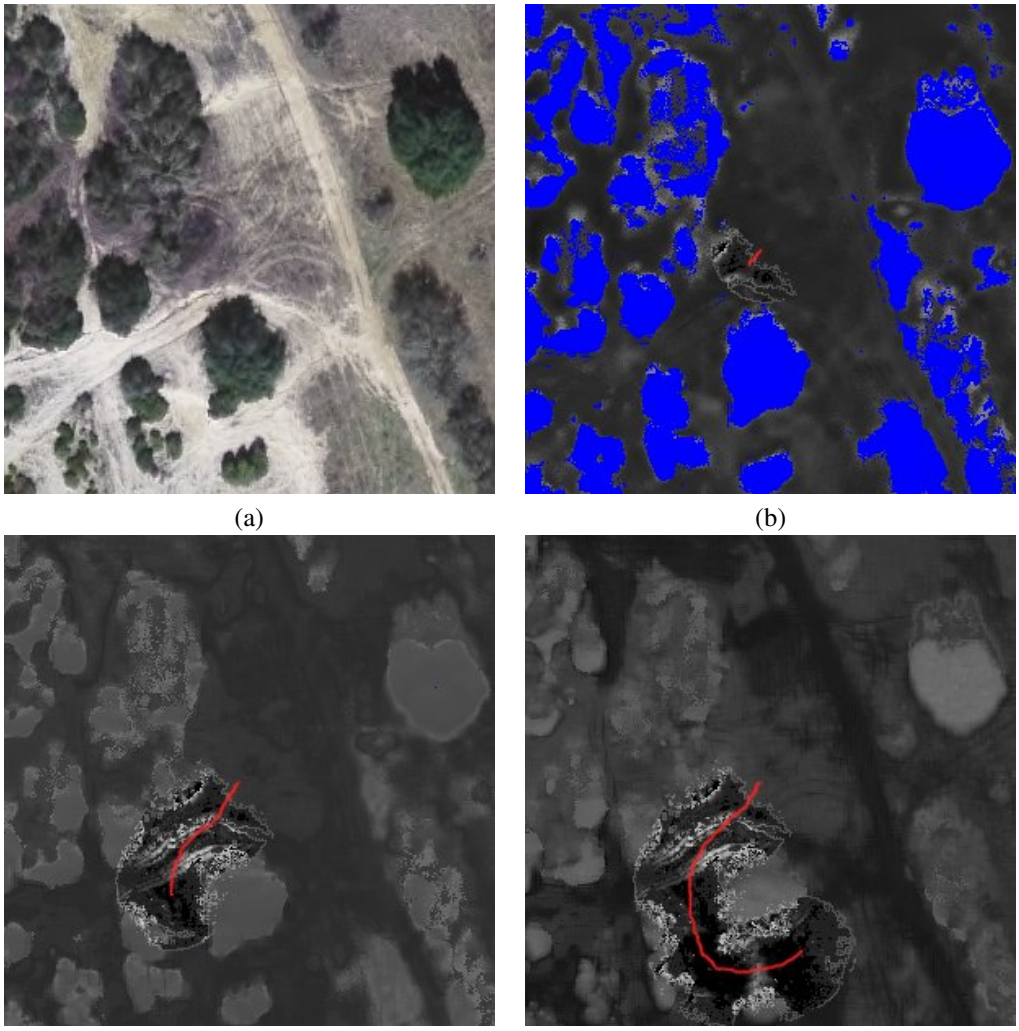
4.3.3 Applications of Trained Algorithm

This algorithm can be used in a variety of ways to aid in unmanned ground vehicle navigation, both in real-time on-board a robot and offline once it has been trained. In the case of online terrain traversal cost prediction, the algorithm can be used to periodically update traversal cost estimates within a region around the robot where features have been computed so that a real-time path planning algorithm such as D* can revise the vehicle's global path to account for the changes. As shown in the following section, the use of this algorithm to extend the robot's field of view results in significantly shorter and more intelligent paths.

When using overhead data, one can make traversal cost predictions for a large area without ever having to traverse or acquire training examples from that area beforehand since the predictive state of the algorithm can be captured at any time by the vector β at that moment. Instead, as long as identical features are computed for the two areas, one can simply drive through a representative area for a short period of time in order to train the algorithm to make predictions in a much larger area. We will also show that a priori traversal cost maps produced by this technique can be more accurate than even those produced from hand-trained techniques that utilize superior data sources.

It is important when using overhead data that the data be aligned with the estimated position of the robot. Even slight mis-registration can significantly hinder the performance of algorithms such as ours that are sensitive to such errors. An advantage of using an online Bayesian linear regression model is the ability to detect and correct such misalignments with relative ease.

When predicting a new traversal cost c_j, the model creates a predictive distribution $p(c)$ with a mean μ_p and a variance σ_p^2. Evaluating the predictive distribution at the traversal cost c_i of a training example gives the probability of having seen that traversal cost given its corresponding feature vector. We can use the probability of having seen all of our data, $p(\tilde{c}_1, \ldots, \tilde{c}_n)$, to detect



(a) (b)

(c) (d)

Figure 4.5: Training progress of online learning algorithm using overhead color imagery data for traversal of environment shown in (a) is shown in (b) - (d). Dimensions of shown areas are 150 m $\times$ 150 m. Accumulated ground truth traversal costs computed by the robot's on-board perception system and vehicle path (shown in red) are overlaid on estimated traversal costs generated by the algorithm. Lower costs appear as darker colors and predictions that the algorithm lacked confidence in (due to insufficient representative training examples) are shown in blue.

map misalignments between overhead data sources and estimated vehicle positions by searching through a space of potential alignments for the one that maximizes the probability of the data.

Since $p(\tilde{c}_1, \ldots \tilde{c}_n)$ can be computed via the chain rule as the product of the predictive distributions evaluated at every $\tilde{c}_i$ used for training, the log data probability is the cumulative sum of $-\log \sigma_p - \frac{(y-\mu_p)^2}{\sigma_p^2}$ for every predictive distribution. After all examples have been received, we compute the average log data probability over all training examples and use this to compare against other alignments. As shown in the following section, correcting such misalignment produces traversal cost maps with better defined obstacles that more accurately reflect the true environment.

Note that for the results in the following section, we chose the hyper-parameter for noise variance σ_l^2 with ML-II and chose an isotropic Gaussian with high variance for the prior on β based on observations from previous robot traversals [37].

This approach allows increased versatility of many UGVs by allowing them to take advantage of data sources that are often ignored while improving performance and removing the necessity of human involvement and parameter engineering.

4.4 Experimental Results

When dealing with overhead data, our algorithm will be referred to as MOLL (Map On-Line Learning)[3] and when dealing with far range sensor data, our algorithm will be referred to as FROLL (Far-Range On-Line Learning). All imagery data was gathered from satellite on average several months prior to traversal and all elevation data was gathered by surveying from helicopter. Both MOLL and FROLL were run on a 1.8 GHz processor with 2 GB of memory. Because of the large amount of aerial data potentially required by MOLL, we implemented a paging system so that only currently relevant areas of overhead data are kept in memory.

4.4.1 Field Test Results

The algorithm was tested with both overhead data and far-range sensor data in real-time on-board our unmanned ground vehicle to measure its impact on navigation performance. 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 series of courses defined by series of waypoints by using only its on-board perception system for navigation. It then traversed the same courses with the help of MOLL, first with 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 and then with FROLL 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 (see Figures 4.6 and 4.7 for sample results). Notice how in Figure 4.6 the MOLL path shows how the robot learned to avoid the dense area of trees after its initial encounter with the area immediately chose to follow the road to the goal.

[3]This algorithm at times has also been referred to as OOLL (Overhead On-Line Learning)

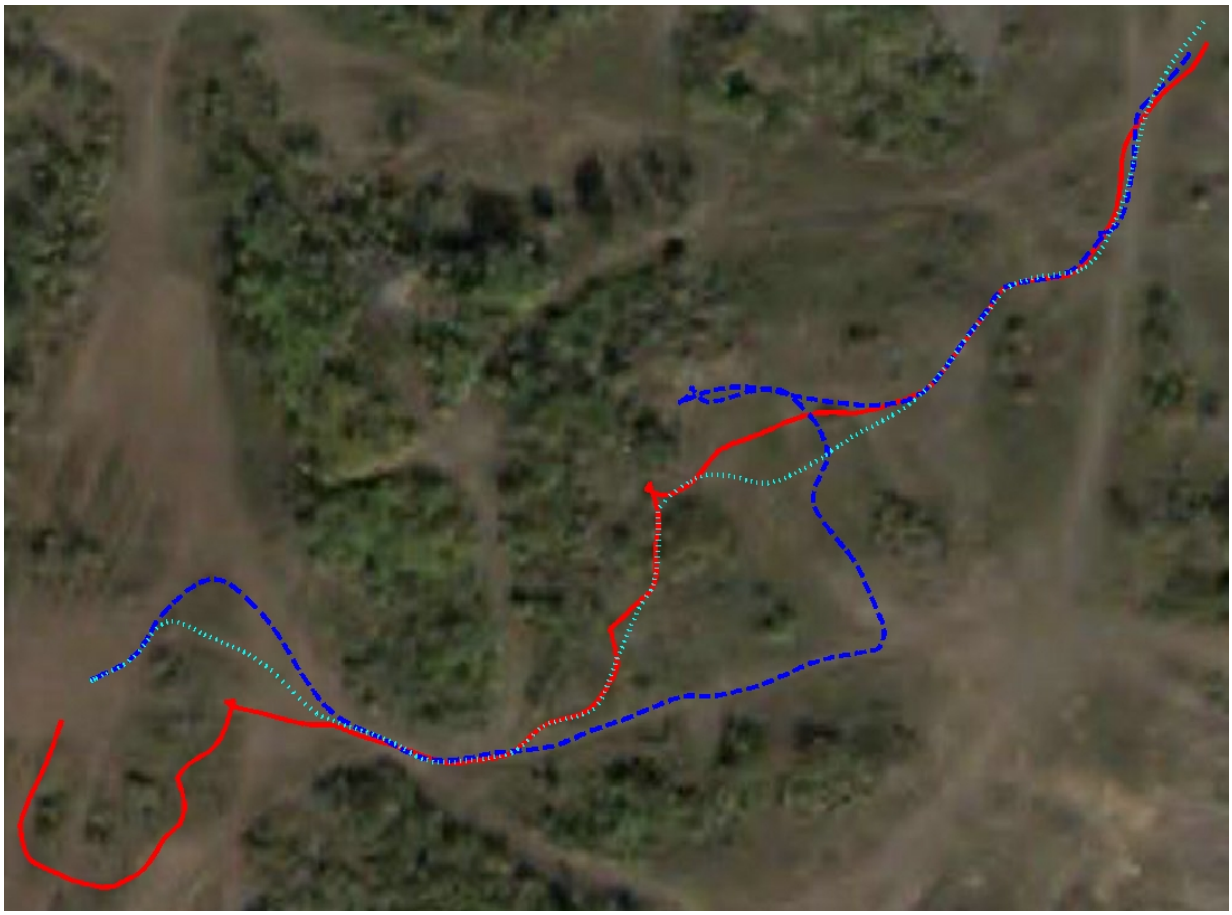


Figure 4.6: Comparison of paths executed by our robot for shown course when using only on-board perception (in solid red), and with MOLL (in dashed blue) and FROLL (in dotted cyan) used in real-time on-board the robot. Course started at the top right and ended at the bottom left.

The FROLL path shows how the robot chose a similar path to the baseline system but was able to give up on dead ends quicker and was able to avoid the large detour at the end of the course due to its extended perception range.

As shown in Table 4.1, our algorithm allowed the vehicle to complete the courses using MOLL in 27% less time while traversing 7% less distance and with FROLL in 34% less time while traversing 13% less distance. Additionally, while we were forced to manually intervene during the tests with only the perception system in order to correct the vehicle's heading when it became trapped in heavy vegetation and could not escape on its own, no manual interventions were necessary when using our algorithm.

While it appears from the shown statistics that FROLL overall resulted in more effective paths than MOLL, we found that with MOLL, the vehicle chose to drive further distances on more preferable terrain in order to avoid difficult or dense areas that presented a larger possibility of damage to the sensors or the need for human intervention. Such an instance an be seen in Figure 4.7a.

Figure 4.7b shows a situation where MOLL can be most useful. Since each traverse began

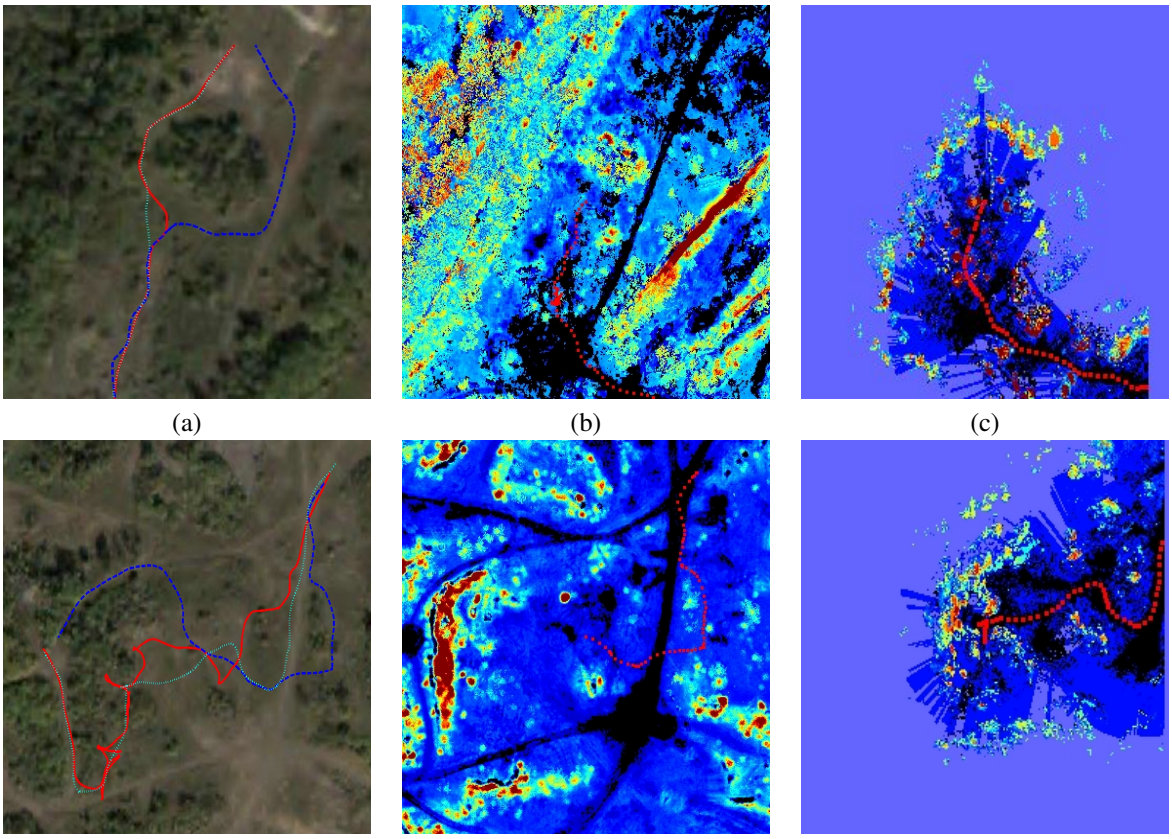


Figure 4.7: Comparison of paths executed for shown situations when using only on-board perception (in solid red) and with MOLL (in dashed blue) and FROLL (in dotted cyan) are shown in (a) and (d). In (a) the course started at the bottom and ended at the top and in (d) the course started at the top right and ended at the left. Predictions of terrain traversal costs for the environment by our algorithm at the times the vehicle chose to avoid the large obstacles in front of it are shown for MOLL in (b) and (e) and for FROLL in (c) and (f). 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 (a) the MOLL path chose to travel slightly further on road in order to avoid the more difficult passage to the left while the FROLL path was able to detect the opening to the left much sooner than the baseline path. In (d) MOLL helped the vehicle avoid the area of dense trees by executing a path that is 43% shorter in 73% less time.

Table 4.1: Statistics for Course Traversals With and Without the Online Learning Algorithm

	Without Algorithm	With MOLL	With FROLL
Total Traversal Time (sec)	1370	1001	898
Total Distance Traveled (m)	1816	1682	1575
Average Speed (m/s)	1.33	1.68	1.75
Number of Interventions	1	0	0

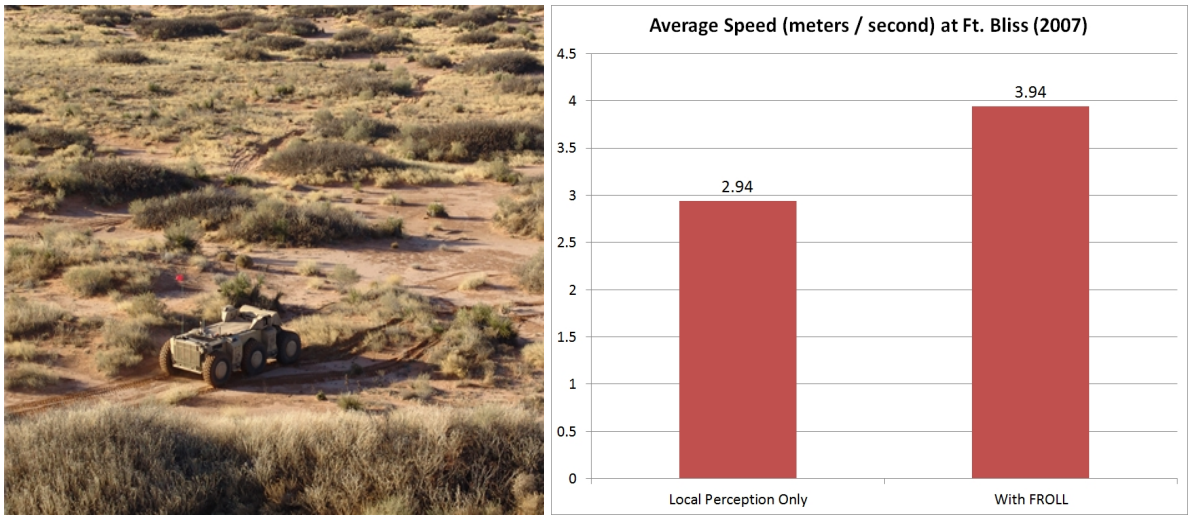


Figure 4.8: The terrain at Fort Bliss is shown at left. Due to the sparseness of the obstacles, using FROLL has a small impact on the total distance traveled, but average speed is significantly increased due to better anticipation of upcoming obstacles (right).

with an untrained algorithm, the MOLL course was given an additional waypoint at the right of the image in order to give it an opportunity to train on a small sample of the environment. As seen in Figure 4.7d, this small amount of training allowed it to identify the wall of trees that heavily hindered the progress of the vehicle in the traverses using only the perception system or FROLL.

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. The demonstrated quantitative impact of these algorithms, however, cannot accurately capture their potential impact on overall performance. Rather, it is important to understand the types of situations that a UGV may encounter for which these algorithms will be most beneficial. If obstacles in an environment are fairly sparse or largely known prior to navigation, extending the range of the perception system will not significantly improve performance metrics. On the other hand, if the environment is full of dense obstacles, hazards and cul-de-sacs with less visible routes towards the goal then the degree of benefit when using such approaches can be arbitrarily large.

The impact of these far-range perception systems depends highly on the environment the robot is operating within. For example, the FROLL system aids navigation in a different way when the environment is more open but has a large number of small isolated obstacles. This type of environment is was encountered at a field test in Fort Bliss in El Paso, Texas in early 2007. A long course through the terrain shown in Figure 4.8 (left image) was executed using only the perception system and then using the FROLL extended perception system. While the total distance traveled using each system was similar due the the sparseness of the obstacles (approximately 3.3 kilometers), the average speed for the system using FROLL was significantly higher due to the ability to detect and avoid obstacles earlier (see Figure 4.8 on right).

FROLL was shown to be a beneficial to autonomous navigation performance and shortly after

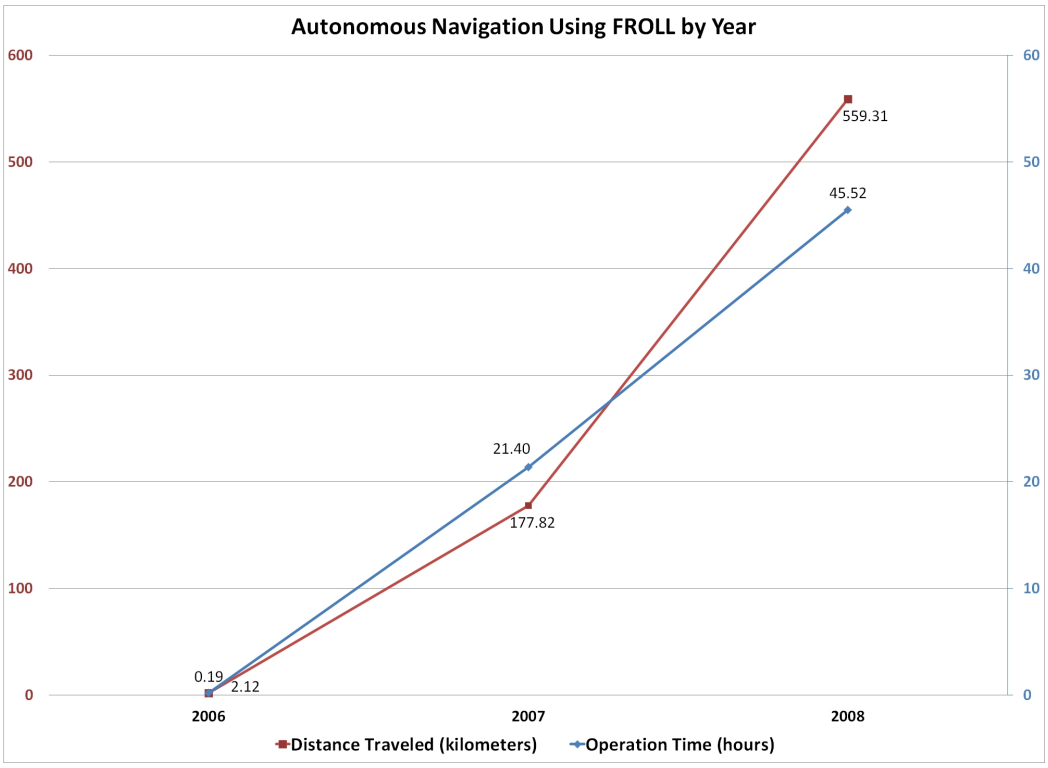


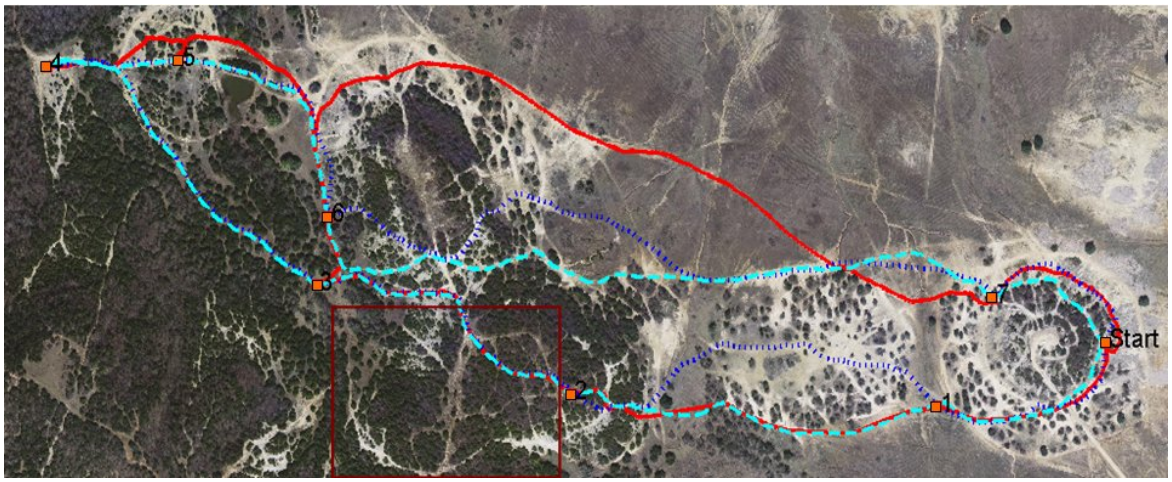
Figure 4.9: After its initial introduction, FROLL quickly became a part of the standard autonomy system used throughout the UPI program. It has been used in official DARPA field tests in over 740 kilometers of autonomous navigation over more than 67 hours.

its initial demonstration it became a standard part of the UPI autonomy system. In late 2007 the FROLL module was rewritten by Cliff Olmstead to make use Velodine laser data. Since initial development in 2006, FROLL was used in official DARPA field tests in over 740 kilometers of autonomous navigation over more than 67 hours (see Figure 4.9), as well as multiples more time and distance throughout development and testing.

4.4.2 Field Test Data Post-Processing Results

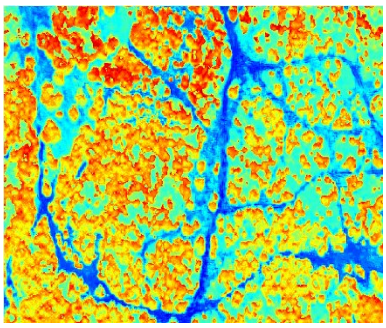
MOLL was also used to produce a priori traversal cost maps for a multi-kilometer course over a large area of complex terrain with heavy vegetation and elevation obstacles defined by a series of GPS waypoints (see Figure 4.10). The algorithm was trained for about 7 minutes using two types of overhead imagery data by driving the vehicle by remote control at about 5 meters per second through the training course outlined by the red box in Figure 4.10a. The trained algorithm was then used off-line to generate a traversal cost map and plan an initial path through the much larger course. Closeups of generated traversal cost maps and resulting planned paths are shown. For comparison, we also included the resulting path from a traversal cost map generated by a supervised learning algorithm with human-picked examples from the actual course and features generated from both overhead imagery and high-density elevation data (see Table 4.2 for a description of data sources) [110].

(a)

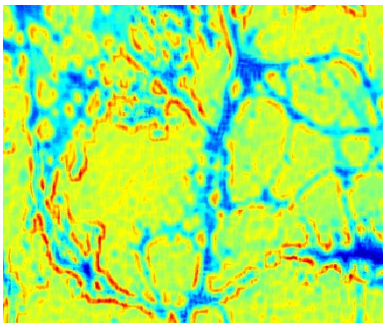


(b)

(c)



(d)



(e)

Figure 4.10: Circular course with the GPS waypoints for which a priori paths were planned is shown in (a). MOLL was trained during a short traversal of the training course outlined in the red box. Shown area is 2000 m $\times$ 750 m. A priori paths generated by a human-trained algorithm (solid red), MOLL using color imagery data (dashed cyan), and MOLL using black and white imagery data (dotted blue) are shown in (b). Traversal cost maps produced by MOLL for the closeup area in (c) using overhead color imagery and black and white imagery are shown in (d) and (e) respectively. See Table 4.2 for description of data sources. Traversal costs are color-scaled for improved visibility where blue and red correspond to lowest and highest traversal cost areas respectively.

Table 4.2: Types of Overhead Data Used by Overhead Online Learning (MOLL) and Hand-Trained Algorithms Used To Produce Prior Cost Maps

Algorithm	Data Used	Resolution
MOLL (color)	Color imagery	0.35 m
MOLL (B & W)	Terraserver B & W imagery	1.0 m
Human-Supervised	Color imagery Elevation	0.35 m < 0.2 m

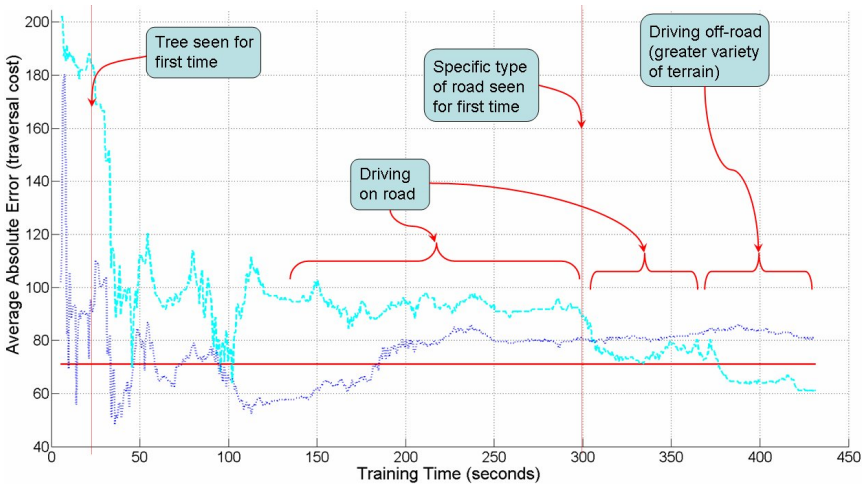


Figure 4.11: Average absolute error between log scale traversal costs computed by the robot's on-board perception system over the course of a multi-kilometer traverse of terrain and traversal cost estimates computed using three techniques: human-trained supervised learning algorithm using high resolution imagery and elevation data (solid red) and MOLL using only color imagery (dashed cyan) and black and white imagery (dotted blue) as a function of training time by driving in a similar environment. See Table 4.2 for a description of data sources. The erratic performance for the first few minutes of training are due to the large effects of new training examples when so little previous data was available. In the case of MOLL with black and white imagery, the initial sample of terrain happened to correspond well to the rest of the course. MOLL training takes longer with color imagery due to a greater number and variety of features.

We evaluated the performance of MOLL against the human-trained technique by accumulating all the traversal costs generated by the vehicle's on-board perception system during a traversal of the course shown in Figure 4.10 and comparing those costs to the estimates from each of the generated prior cost maps. The average absolute error in traversal cost (on a log scale as described earlier) for each method is shown in Figure 4.11 as a function of training time. This result shows that MOLL is competitive with respect to the human-trained algorithm using only imagery data after only a short period of training. However, it should be pointed out that maintaining a tight correspondence from traversal costs assigned by the human-trained algorithm to those assigned by the perception system was difficult to strictly enforce. This highlights another advantage of the online learning approach over a human-trained approach: by relieving the need for manual manipulations of traversal cost assignment strategies, the entire system is more adaptable to changes in both the environment and the perception system.

During post-analysis of this test, we discovered that the overhead imagery data and the esti-

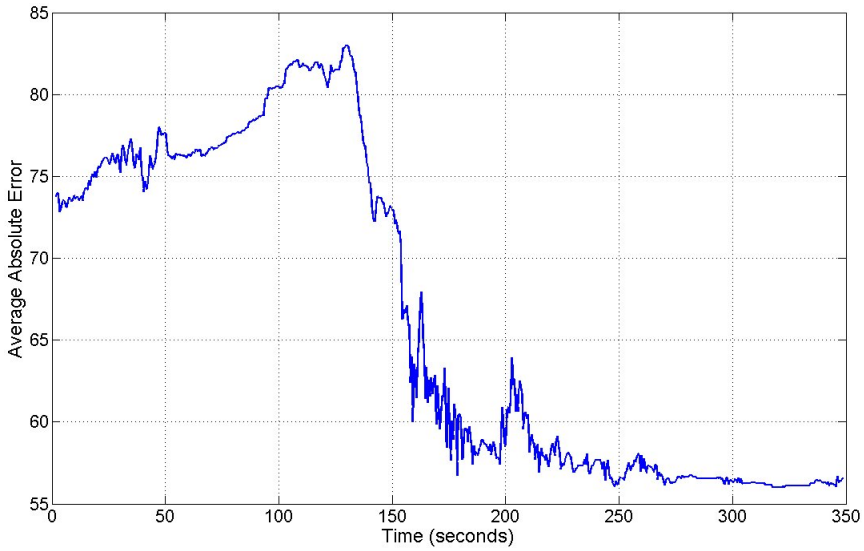


Figure 4.12: Average absolute error between log scale traversal costs computed by the robot's on-board perception system and traversal cost estimates generated by FROLL as a function of training time.

mated position of the vehicle were in fact misaligned by about 1.5 m. While this result shows that our algorithm is robust to such map misalignment, this article also demonstrates how our algorithm can be used to detect such errors in alignment in order to achieve optimal performance.

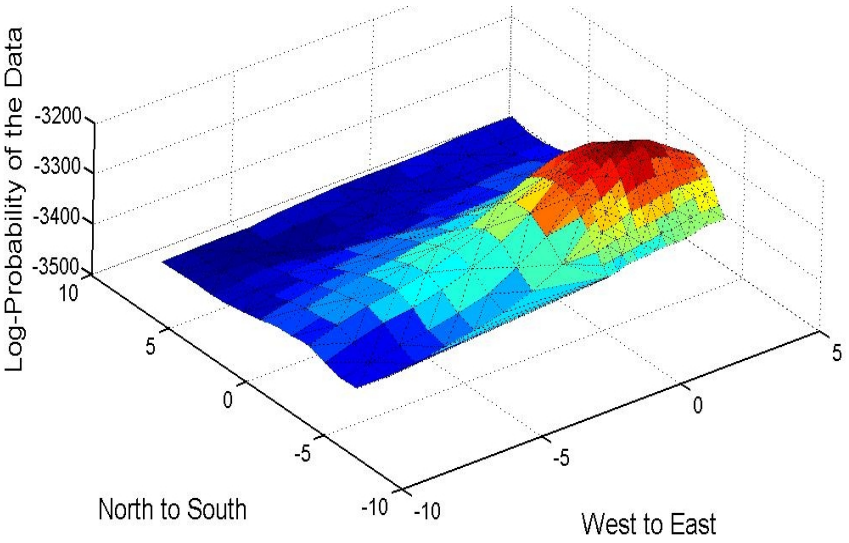
Performance of FROLL was similarly evaluated by comparing its estimates to all computed perception costs during a course as a function of training time. Only estimates that had not been used for training yet were included in this calculation (so as not to use the training set for testing). The results can be seen in Figure 4.12. Notice how after only three minutes of training the algorithm has mostly converged to its final predictive performance.

4.4.3 Offline Map Alignment

We applied our map alignment technique to a manually misaligned log of perception data and overhead imagery features. A brute force search across all potential map alignments in 0.35 m increments in the four cardinal directions detected a misalignment of 3.85 m west and 4.9 m north. Such a search is too computationally expensive to be performed in real-time but was completed in about an hour through offline processing. Computed probabilities of observed perception data and the corresponding improvement in traversal cost estimates can be seen in Figure 4.13. As expected, correcting the misalignment improved the definition of obstacles in the traversal cost maps and resulted in a stronger correspondence with the actual environment, correctly showing that the traveled path is clearly on the road.

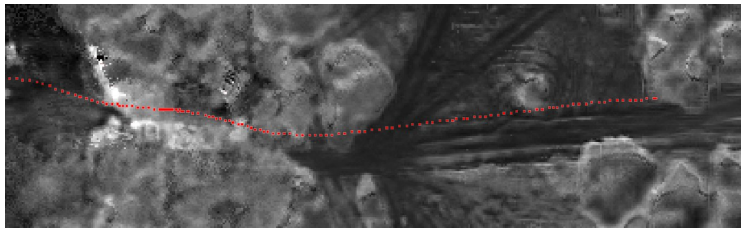
4.4.4 Feature Selection

We applied our feature selection imagery to a set of 33 overhead imagery and elevation data based features. While these features were all relevant to the environment, we assumed that many

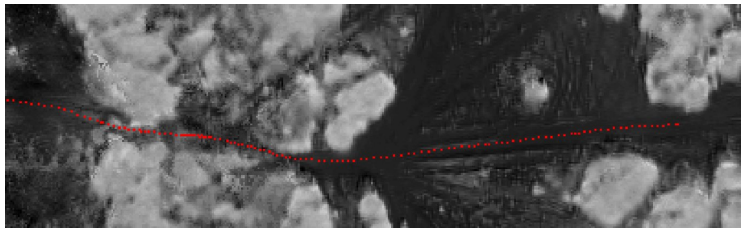


(a)

(b)



(c)



(d)

Figure 4.13: Example of how misalignment between overhead data sources and estimated vehicle position can be detected using our algorithm. Computed log probability of the perception system sensor data encountered over a 12.6 m $\times$ 12.6 m search space of alignment shifts is shown in (a). MOLL cost prediction for area shown in (b) before alignment correction and after correcting detected misalignment of 3.85 m west and 4.9 m north) appear in (c) and (d) respectively (best alignment is assumed to be that which produces the highest probability of seen perception data). Darker colors in the images correspond to lower traversal costs. The robot's path is shown in red.

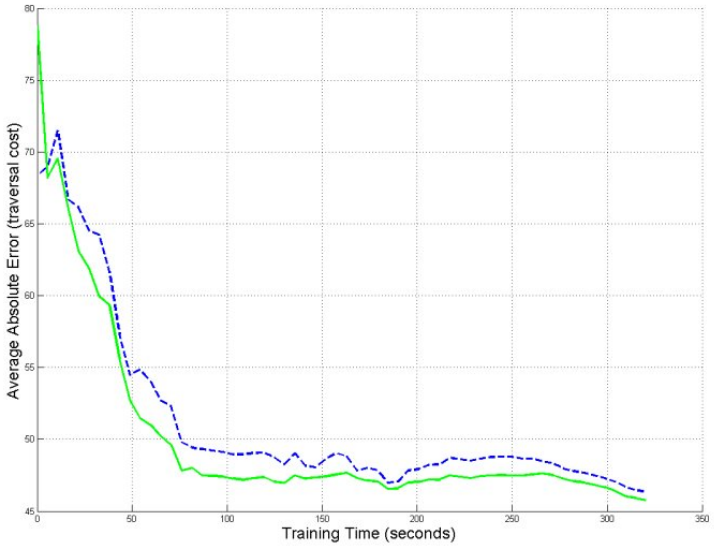


Figure 4.14: Average absolute error between log scale traversal costs computed by unmanned ground vehicle's on-board perception system over the course of a multi-kilometer traversal of terrain and traversal cost estimates computed before and after feature selection: results of using the full set of 33 features from both imagery and elevation data (dotted blue) and a subset of 14 features selected using the feature selection algorithm (solid green) are shown.

of them were redundant and therefore selected the 14 most important from the set. Figure 4.14 depicts the accuracy of cost predictions as a function of training time using this reduced set of features compared to the original set. As expected, the smaller set of selected features resulted in a decreased training time.

Chapter 5

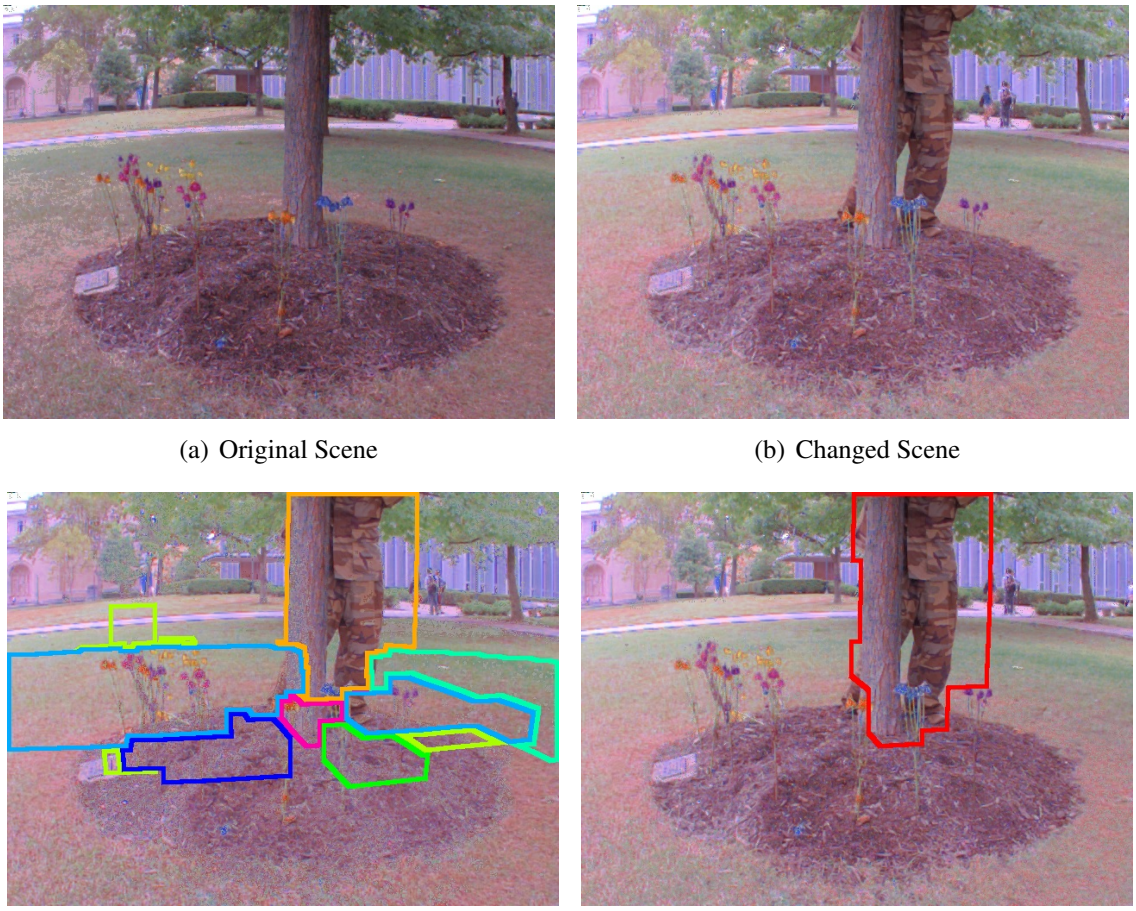
Anytime Online Novelty and Change Detection

Mission-ending mistakes are a key concern in deploying mobile robots. One approach to facilitating safe traversal is for a UGV to be able to identify situations that it is likely untrained to handle *before* it experiences a major failure. This problem therefore becomes one of novelty detection: how a robot can identify when perception system inputs differ from prior inputs seen during training or operation. 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.

Change detection is a closely related (though less studied) problem to novelty detection where a robot needs to operate repeatedly in an environment and must detect when a significant change has occurred from a previous operation. Such changes could include the appearances of obstacles such as cars, barriers, fallen trees or humans. A reliable change detection system allows a robot to rely on the fact that if it had successfully navigated through an area multiple times in the past, the most likely potential for hazards comes from changes to the environment.

We pursue this problem of change detection as a location-specific version of novelty detection where we detect novelty of locations in the current scene with respect to those same locations in a past navigation (rather than with respect to all prior experiences as we would with novelty detection). Using such a formulation allows us to address both problems using algorithms that are at their core quite similar. Furthermore, we present an online scene segmentation algorithm that when used in conjunction with our change detection system can lead to significantly more accurate performance across diverse environments. When operating frequently in the same area, a robust and reliable change detection system can safeguard both the vehicle and the environment (see Figure 5.1 for an example use).

Two common limitations of novelty detection systems are particularly relevant to the mobile robotics domain. Autonomous systems often need to learn from their experiences and continually adjust their models of what is normal and what is novel. For example, if human feedback were to confirm that a certain type of environment selected as novel is actually safe to handle with the existing autonomy system or demonstrate to the system the proper way to handle the situation (as in [112]), the model no longer needs to identify such inputs as novel. Most novelty detection approaches, however, build a model of the normal set of examples a priori in batch in order to



(a) Original Scene

(b) Changed Scene

(c) Real-time Segmentation Result

(d) Detected Changes

Figure 5.1: An example of the change detection system in use. In 5.1(b) a change to the original scene shown in 5.1(a) is introduced. The result of the online scene segmentation system (shown in 5.1(c)) are helpful in reducing false positives, allowing more accurate detection of introduced changes (shown in 5.1(d)).

detect novel examples in the future but are unable to update that model online without retraining.

Furthermore, existing novelty detection techniques see diminished performance when using high-dimensional feature spaces, particularly when some features are less relevant, redundant, or noisy. These qualities are particularly common in features from many UGV perception systems due to the variety of sensors and uncertainty about how these features relate to novelty. For example, a large variety of camera-based features from color and texture might be computed for use in various components of the perception system. While these features are potentially powerful, subsets of these features often contain redundant irrelevant information. It is therefore important for novelty detection techniques to be resilient to such feature properties.

We present an online approach that addresses these common problems with novelty (and change) detection techniques. In order to deal with the diversity of possible data, we approach these problems with a kernel machine that we use to make estimates of similarity. Because we want seen examples to generate an influence of familiarity in feature space toward future exam-

ples, a big challenge becomes identifying a feature space for operation that has the property that proximity implies similarity. When prior class information is available, we show how using Multiple Discriminant Analysis (MDA) for generating a reduced dimensional subspace to operate in rather than other common techniques such as Principal Components Analysis (PCA) can make the novelty and change detection system more robust to issues associated with high-dimensional feature spaces. In effect, this creates a lower dimensional subspace that truly captures *what makes things novel*.

Additionally, our algorithm can be framed as a variant of the NORMA algorithm, an online kernelized Support Vector Machine (SVM) optimized through stochastic gradient descent, and therefore shares its favorable qualities [61]. However, by incorporating a simple data structure optimization step, our algorithm greatly improves its ability to maintain an accurate model over long periods of operation while benefiting from its anytime properties and strong bounds on necessary computation time.

While this work was targeted toward mobile robotics applications, the approaches here are more generally applicable to any domain which has similar requirements.

The next section presents background on novelty and change detection techniques and some example applications. Section 5.2 presents our novelty detection algorithm and Section 5.3 shows how it can be extended to be able to perform change detection. Following that is an explanation of the online scene segmentation system in Section 5.4, Section 5.5 explains how these techniques can be applied to mobile robotics and field test results from several large UGVs are presented in Section 5.6.

5.1 Related Work

Techniques dealing with novelty detection (also referred to as anomaly or outlier detection) problems and the closely related change detection problem have been applied to a wide range of domains such as detecting structural faults [145], abnormal jet engine operation [45], computer system intrusion detection [100], and identifying masses in mammograms [127]. In the robotics domain some have incorporated novelty detection systems within inspection robots [81, 91].

Novelty detection is often treated as a one-class classification problem. In training the system sees a variety of "normal" examples (and corresponding features) and later the system tries to identify input that does not fit into the trained model in order to separate novel from non-novel examples. Instances of abnormalities or novel situations are often rare during the training phase so a traditional classifier approach cannot be used to identify novelty in most cases.

Most novelty detection approaches fall into one of several categories. Statistical or density estimation techniques model the "normal" class in order to identify whether a test sample comes from the same distribution or not. Such approaches include Parzen window density estimators, nearest neighbor-based estimators, and Gaussian mixture models [29]. These techniques often use a lower-dimensional representation of the data generated through techniques such as PCA.

Other approaches attempt to distinguish the class of instances in the training set from all other possible instances in the feature space. Schölkopf et al. [103] show how an SVM can

be used for specifically this purpose. A hyper-plane is constructed to separate the data points from the origin in feature space by the maximum margin. One application of this technique was document classification [78]. A noticeable drawback of this approach is that it makes an inherent assumption that the feature space origin is a suitable prior for the novel class. This limitation was addressed by [15] by attracting the decision boundary toward the center of the data distribution rather than repelling it from the origin. A similar approach encloses the data in a sphere of minimal radius, using kernel functions to deal with non-spherically distributed data [128]. These techniques all require solutions to linear or quadratic programs with slack variables to handle outliers.

Another class of techniques attempts to detect novelty by compressing the representation of the data and measuring reconstruction error of the input. The key idea here is that instances of the original data distribution are expected to be reconstructed accurately while novel instances are not. A simple threshold can then be used to detect novel examples. The simplest method of this type uses a subset of the eigenvectors generated by PCA to reconstruct the input. An obvious limitation here is that PCA will perform poorly if the data is non-linear. This limitation was addressed by using a kernel PCA based novelty detector [48]. Benefits of more sophisticated auto-encoders, neural networks that attempt to reconstruct their inputs through narrow hidden layers, have been studied as well [53].

Online novelty detection has received significantly less attention than its offline counterpart. Since it is often important to be able to adjust the model of what is considered novel in real-time, many of the above techniques are not suitable for online use as they require significant batch training prior to operation. While Neto et al. [91] replaced the use of PCA for novelty detection with an implementation of iterative PCA, performance was still largely influenced by the initial data set used for training. Marsland proposed a unique approach that models the phenomenon of habituation where the brain learns to ignore repeated stimuli [81]. This is accomplished through a clustering network called a Grow When Required (GWR) network. This network keeps track of firing patterns of nodes and allows the insertion of new nodes to allow online adaptation.

Markou and Singh have written a pair of extensive survey articles detailing many additional novelty detection applications and techniques [79, 80].

While change detection is less frequently studied directly than novelty detection, many novelty detection approaches (including those discussed above) can be adapted for change detection problems (or are already performing change detection). An obvious application of change detection is to the problem of real-time surveillance. Several researchers in the computer vision community detect changes in video streams using *temporal difference* methods [51, 77, 95]. Another approach simply compares each current image with the same background at a previous time [67]. These image-based techniques are obviously sensitive to varying viewpoints and variations in lighting.

Others have applied change detection techniques to analyzing land-cover changes over long periods of time. These include the analysis of forest defoliation [87], reductions of tropical forests [133], and an analysis of land use over time [118]. A quality survey of various change detection techniques for monitoring land-cover changes can be found in [82].

One of the more rigorous studies on change detection in the mobile robotics domain can be

found in [96]. Here the authors approached the problem by directly comparing features computed by the perception system, attempting to identify changes from variations within those individual features. This approach had two significant limitations. First, in order to perform these direct comparisons, it relied on a highly-accurate positioning system whose error was at almost all times less than 20 cm, an unrealistic expectation for a majority of systems utilizing GPS technologies. Also, because the system relied on individual features, the final system was heavily tuned to use certain features over others and varied the sensitivity manually for each. The strict registration requirements and sensitivity to features resulted in a large number of false positives and would be a difficult approach to generalize to other scenarios.

One of the main limitations that the above-mentioned novelty and change detection approaches share is a quick deterioration in performance as the number of less relevant or noisy features grows. The disproportionately high variance of many of these features make it difficult for many of these algorithms to capture an adequate model of the training data and their effects quickly begin to dominate more relevant features in making predictions. Our algorithm addresses this crucial limitation in cases where class information is available within the training set while still being suitable for online use.

5.2 Approach

5.2.1 Formalization

The goal of novelty detection can be stated as follows: given a training set $D = \{\mathbf{x}\}_{1...N} \in \mathcal{X}$ where $\mathbf{x}_i = \{x_i^1, \ldots, x_i^k\}$, learn a function $f : \mathcal{X} \rightarrow \{novel, not\text{-}novel\}$. In the online scenario, each time step t provides an example $\mathbf{x}_t$ and a prediction $f_t(\mathbf{x}_t)$ is made.

Algorithm 2 Online novelty detection algorithm

given: A sequence of features $S = (\mathbf{x}_i)_{1...T}$; a novelty threshold γ; a learning rate η
outputs: A sequence of hypotheses $\mathbf{f} = (f_1(\mathbf{x}_1), f_2(\mathbf{x}_2), \ldots)$
for $t = 1$ to T **do**
  $f_t(\mathbf{x}_t) \leftarrow \sum_{i=1}^{t-1} \alpha_i k(\mathbf{x}_i, \mathbf{x}_t)$
  if $f_t(\mathbf{x}_t) < \gamma$ **then**
    $\alpha_t \leftarrow \eta$
  else
    $\alpha_t \leftarrow 0$
  end if
end for

We perform online novelty detection with a kernel machine that we use to make estimates of similarity as shown in Algorithm 2 [117]. All possible functions f are elements of a *reproducing kernel Hilbert space* $\mathcal{H}$ [102]. All $f \in \mathcal{H}$ are therefore linear combinations of kernel functions:

$$f_t(\mathbf{x}_t) = \sum_{i=1}^{t-1} \alpha_i k(\mathbf{x}_i, \mathbf{x}_t) \tag{5.1}$$

We make the assumption that proximity in feature space is directly related to similarity. Observed examples deemed as novel are therefore remembered and have an influence of familiarity on future examples through the kernel function $k(\mathbf{x}_i, \mathbf{x}_j)$. A novelty threshold, γ, and a learning rate, η, are initially selected. For each example $\mathbf{x}_t$, the algorithm accumulates the influence of all previously seen novel examples (line 4). If this sum does not exceed γ then the example is identified as novel and is remembered for future novelty prediction (line 6)[1]. Non-novel examples are not stored as they are assumed to have minimal impact on future novelty computations (even though a coefficient of 0 is assigned in line 8 for clarity, these examples are not stored). We suggest simply using the Gaussian kernel with an appropriate variance σ^2:

$$k(\mathbf{x}_i, \mathbf{x}_j) = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma^2}} \tag{5.2}$$

This algorithm assumes that every seen query is novel until it can be proven otherwise. It is therefore important to understand that the threshold of γ in line 5 is the point at which a query becomes non-novel. Only those examples that are similar to the query (those that contribute toward this threshold) will therefore impact the final decision.

5.2.2 Improved Dimensionality Reduction

Especially if the number of features is large, it may first be necessary to project the high-dimensional input $\mathbf{x}_t$ into a lower-dimensional subspace more suitable for novelty detection using distance metrics. The big challenge becomes how to construct this feature transformation so that this space has the property where proximity implies similarity.

PCA, the most common approach for this purpose among dimensionality reduction (and novelty detection) techniques, finds a linear transformation that minimizes the reconstruction error in a least-squares sense. If subsets of the features are redundant, noisy or are dominated disproportionately by a subset of the training set, however, applying techniques such as PCA, or any other unsupervised dimensionality reduction technique, may yield disappointing results as precisely the most relevant directions for differentiation may be discarded in order to reduce reconstruction error of a less relevant portion of the feature space.

Rather than optimizing for reconstruction error, *discriminant analysis* seeks transformations that are efficient for discriminating between different classes within the data. Multiple Discriminant Analysis (MDA), a generalization of Fischer's linear discriminant for more than two classes, computes the linear transformation that maximizes the separation between the class means while keeping the class distributions themselves compact, making it useful for classification tasks [29].

We argue that when prior class information for the training set is available, using MDA to construct a lower dimensional subspace using labeled classes not only optimizes for known class

[1]While both the novelty threshold, γ, and the learning rate, η, are included for clarity, they only represent one degree of freedom since the equations can be simply rescaled to be functionally equivalent with $\eta = 1$.

separability but likely leads to separability between known classes and novel classes. In cases described earlier that result in poor performance when using PCA, MDA will largely ignore features that do not aid in class discrimination, instead focusing on the strongly differentiating features.

Novelty detection is about encountering new classes, so by using discriminating ability as the metric for constructing a subspace, one can capture the combinations of features that make known classes novel *with respect to each other* and likely generalize to previously unseen environments, in effect capturing *what makes things novel*. We are in effect performing transfer learning here where we learn an optimal kernel function from one task in order to perform well on a related task.

Additionally, we can take advantage of the labeled prior class data that we used for training to choose an appropriate σ by finding the value that optimizes the ratio between inter-class contribution to outer-class contribution for a random subset of examples.

While this algorithm could be performed in other domains using the raw features, from this point forward $\mathbf{x}_t$ will refer to the projection of the raw features into the lower dimensional space rather than the raw features themselves. Experimental validation of this theory within the domain of mobile robotics is presented in Sections 5.5 and 5.6.

5.2.3 Framing as Instance of NORMA

The NORMA algorithm is a stochastic gradient descent algorithm that allows the use of kernel estimators for online learning tasks [61]. As with our algorithm, f is expressed as a linear combination of kernels (see equation 5.1). NORMA uses a piecewise differentiable convex loss function l such that at each step t we add a new kernel centered at $\mathbf{x}_t$ with the coefficient:

$$\alpha_t = -\eta l'(\mathbf{x}_t, y_t, f_t) \tag{5.3}$$

Our algorithm can easily be framed as an online SVM instance of NORMA using a hinge loss function as follows:

$$y_t = \gamma \tag{5.4}$$
$$l(\mathbf{x}_t, y_t, f_t) = max(0, y_t - f_t(\mathbf{x}_t)) \tag{5.5}$$

Taking the subgradient [109] of (5.5), we get:

$$l'(\mathbf{x}_t, y_t, f_t) = \begin{cases} -1 & \text{if } f_t(\mathbf{x}_t) < \gamma \\ 0 & \text{otherwise} \end{cases} \tag{5.6}$$

As before, the gradient of our loss is non-zero only when the accumulated contributions from stored examples are less than the novelty threshold γ, signifying that the example is novel. From (5.3) and (5.6) we then get:

$$\alpha_t = \begin{cases} \eta & \text{if } f_t(\mathbf{x}_t) < \gamma \\ 0 & \text{otherwise} \end{cases} \tag{5.7}$$

This is equivalent to the update steps in lines 6 and 8 of Algorithm 2, showing that our algorithm can be framed as a specific instance of the NORMA algorithm which produces a variety of useful bounds on the expected cumulative loss [61].

5.2.4 Query Optimization

Without further measures, the potential number of basis functions stored by Algorithm 2 could grow without bound. NORMA deals with this issue by decaying all coefficients α_i and dropping terms when their coefficients fall below some threshold. We found that to prune in this way to an effective fixed capacity, the decay rate has to be relatively high which leads to a forgetting effect. The qualitative effect is that examples that were previously encountered are re-flagged as novel which would result in unnecessary human interventions.

Instead, we propose in Algorithm 3 a modified anytime version of Algorithm 2 that better utilizes a fixed buffer size while ensuring efficient and bounded computation. This algorithm takes advantage of the fact that the familiarity contribution to new queries is often dominated by only a few examples. First, we can easily gain some efficiency by only processing stored examples until we have reached the novelty threshold (line 7).

The key performance improvement comes from the sequence optimization step in line 17. For each prediction, the stored example that breaks the novelty threshold γ, or the new novel example itself, is moved to the front of the list as it is more likely to impact future queries.

In addition to allowing us to bound the number of stored examples (line 19) and perform favorably with respect to NORMA, this gives our algorithm an anytime property by enabling it to quickly classify as much of the environment as possible as not novel. When this algorithm is unable to run to completion due to time constraints, it will fail intelligently by generating false positives but never potentially dangerous false negatives.

This is a slight variation of the traditional problem of dynamically maintaining a linear list for search queries for which the move-to-front approach was proven to be constant-competitive, meaning no algorithm can beat this approach by more than a constant factor [113]. This approach works extremely well in practice because a large majority of queries are *not* novel, meaning that we are racing to reject them as quickly as possible. By continually adjusting the list in this way, we are able to adapt to changing environments while maintaining fast operation times.

A rigorous discussion of this list maintenance problem and a variety of potential strategies can be found in Appendix B.

5.3 Extension to Change Detection

In many scenarios a robot must operate repeatedly in the same area. Examples of such domains include construction, mining, patrolling, or supply routes. If the robot has navigated through an environment multiple times successfully before, it is likely do to so again if the environment remains unchanged. Therefore, the biggest indicator of a potential hazard is when something in the environment has changed. For example, such changes could include a vehicle or barrier

Algorithm 3 Online novelty detection algorithm with query optimization

1: **given:** A sequence of features $S = (\mathbf{x}_i)_{1...T}$; a novelty threshold γ; a learning rate η; a maximum example storage capacity N
2: **outputs:** A sequence of hypotheses $\mathbf{f} = (f_1(\mathbf{x}_1), f_2(\mathbf{x}_2), \ldots)$
3: **initialize:** $n \leftarrow 0$
4: **for** $t = 1$ to T **do**
5: $\quad i \leftarrow 1$
6: $\quad f_t(\mathbf{x}_t) \leftarrow 0$
7: $\quad$ **while** $f_t(\mathbf{x}_t) < \gamma \; and \; i \leq n$ **do**
8: $\quad\quad f_t(\mathbf{x}_t) \leftarrow f_t(\mathbf{x}_t) + \alpha_i k(\mathbf{x}_i, \mathbf{x}_t)$
9: $\quad\quad i \leftarrow i + 1$
10: $\quad$ **end while**
11: $\quad$ **if** $f_t(\mathbf{x}_t) < \gamma$ **then**
12: $\quad\quad \alpha_{n+1} \leftarrow \eta$
13: $\quad\quad \mathbf{x}_{n+1} \leftarrow \mathbf{x}_t$
14: $\quad\quad n \leftarrow n + 1$
15: $\quad\quad i \leftarrow i - 1$ $\quad$ {i was incremented one extra time}
16: $\quad$ **end if**
17: $\quad$ **optimize sequence:** Move $(\alpha_i, \mathbf{x}_i)$ to front
18: $\quad$ **if** $n > N$ **then**
19: $\quad\quad$ Delete $(\alpha_i, \mathbf{x}_i)_{i>N}$ {crop to list size N}
20: $\quad\quad n \leftarrow N$
21: $\quad$ **end if**
22: **end for**

At line 17, if $f_t(\mathbf{x}_t) =$ *not-novel*, i indexes the example that broke the novelty threshold. Otherwise, i indexes $\mathbf{x}_t$.

appearing in the road, significant vegetation growth, or most importantly, a human entering the scene.

This is an exceptionally difficult problem for several reasons. First, the system must be able to deal with natural variations in the environment from one traversal to the next, including slight variations in paths driven and resulting variations in sensing angles and density. Such changes would obviously cause variations in computed features between the two traversals that must be filtered to avoid excessive false positives. Also, such techniques must be able to handle small amounts of registration error typical of real-world scenarios. The approach described in this section is able to handle both challenges while maintaining low rates of false positives.

We treat the problem of change detection as a location-specific instance of novelty detection. Rather than trying to identify situations that are novel with respect to everything seen previously, a change detection system needs to identify when a situation is novel with respect to how the situation looked at an earlier time (from a stored log for example). To accomplish this we use a compact instance of Algorithm 2 at each location in the visible scene and detect novelty against a stored representation of the scene at a previous time.

Algorithm 4 Online change detection algorithm (modification of Algorithm 2)

1: **given:** A set of voxel features $(\mathbf{x}_i)_{1\ldots N}$ from a location in the current scene; a sequence of corresponding voxel features from within a specified radius r of that location from a previous navigation $S = (\tilde{\mathbf{x}}_j)_{1\ldots M}$; a novelty threshold γ
2: **outputs:** A sequence of hypotheses $\mathbf{f} = (f_1(\mathbf{x}_1), f_2(\mathbf{x}_2), \ldots, f_N(\mathbf{x}_N))$ where $f_i(\mathbf{x}_i)$ specifies if voxel $\mathbf{x}_i$ has changed from the previous iteration
3: **for** $i = 1$ to N **do**
4: $\quad y \leftarrow 0$
5: $\quad y \leftarrow \sum_{\tilde{\mathbf{x}}_j \in S} k(\mathbf{x}_i, \tilde{\mathbf{x}}_j)$
6: $\quad$ **if** $y < \gamma$ **then**
7: $\quad\quad f_i(\mathbf{x}_i) \leftarrow$ true {This voxel has changed}
8: $\quad$ **else**
9: $\quad\quad f_i(\mathbf{x}_i) \leftarrow$ false {This voxel is un-changed}
10: $\quad$ **end if**
11: **end for**

Algorithm 4 shows this slight modification of the novelty detection algorithm shown in Algorithm 2 for use in change detection. We assume a log of all seen voxel features is available from a previous traversal of the environment (only the lower-dimensional representation of the features needs to be stored since comparisons happen in the MDA-defined space rather than the raw feature space).

For each seen voxel, the kernel function is used to sum the measures of similarity of that voxel with respect to all voxels near that location from the previous traversal (line 5). If the example is novel with respect to that set of voxels, then the system specifies that the voxel has changed from before. All voxels in some region around the example are used to allow the algorithm to be robust to moderate amounts of registration error between the two data sets.

Because the addition of any irrelevant voxels provides little contribution toward the novelty

threshold in the final prediction (only similar voxels will contribute to the final measure), the addition of these extra buffer voxels does not influence the prediction in a majority of scenarios. For example, if a change has been introduced into the scene, using a slightly larger footprint for comparison from the previous traversal would still result in a change being detected because the extra voxels would only help the query reach the novelty threshold γ in the unlikely case where they were extremely similar to the changed object itself. Further discussion on dealing with registration error can be found in Section 5.6.2.

Notice that the query optimization improvement from Algorithm 3 is not necessary here because each instance of novelty detection will only be used once and will operate on only a small number of examples.

5.4 Improving Performance through Scene Segmentation

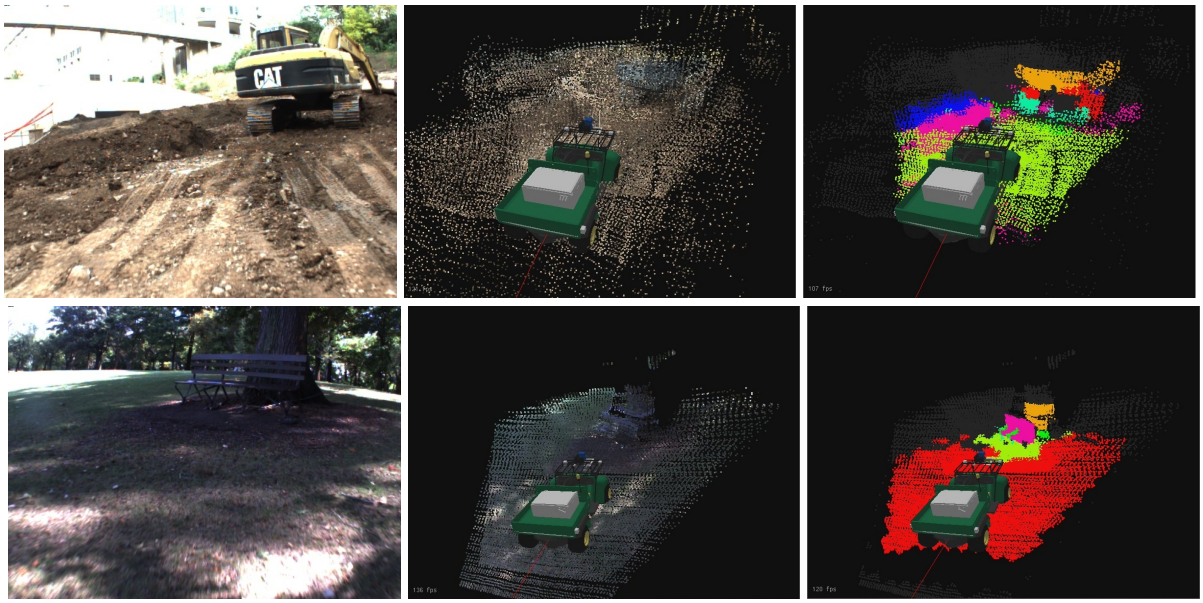


Figure 5.2: Example segmentation results for several scenes. The left, center, and right images show the camera view, 3-D laser data, and final segmentations respectively.

The approach to change detection described in Section 5.3 is analogous to a sliding window approach for image classification. Evaluating one voxel at a time works for many situations, but losing the ability to reason contextually about the scene leads to many false positives in the same way that classifying objects in an image is difficult when only considering one pixel or small region at a time. We now propose an improvement to the change detection system described earlier that first performs an online segmentation of the scene and then uses that information within the voxel-based change detection system to better analyze the environment. Example segmentation results can be seen in Figure 5.2.

This system learns and makes use of a general "similarity" classifier to iteratively segment the scene, allowing us to deal with any number or segments and previously unseen objects. This

segmentation module adds contextual information to the change detection process, increasing accuracy and reliability as shown in the example in Figure 5.2. Although the same extension could improve the novelty detection system as well, we will focus here only on its impact on the change detection system.

The problem of segmenting data into meaningful regions has been extensively researched in the computer vision domain [22, 32, 107, 108], and to a lesser degree in the context of 3D data. In the case of 3D segmentation, a scene is often represented with a graphical model connecting the 3D components (in our case voxels). A basic approach is to segment based purely on 3D geometry [41], but this approach is obviously limited in that it considers only spatial dimensions and ignores valuable feature information.

Others have worked to create basic classifiers from feature-rich 3D scenes that can segment a scene into vegetation, ground, and non-traversable obstacles [89]. It performed well on the classes of objects with which it was trained, but does not generalize to general-purpose segmentation in unknown environments. More recently, some approaches have tried to directly detect known objects in the scene [50, 54] or detect and analyze shapes inside point clouds [86, 99], but this is again unable to deal with previously unknown object classes.

Others have tried to create a discriminative 3D segmentation technique which uses a subclass of Markov Random Fields that provided impressive experimental results [3]. Their approach is not entirely applicable to change detection because they require training data which contains representative objects, and then they use segmentation to detect those same types of objects in point clouds.

Unlike many of the previously-mentioned approaches, our approach is able to generalize to new environments because it trains and utilizes a classifier which operates on any two voxels' features to predict whether they are in the same class or not, without explicitly requiring knowledge about the various classes themselves. Similar to our use of MDA for dimensionality reduction, this classifier also learns to identify the core combinations of features that make objects different with respect to each other, regardless of their class.

5.4.1 Segmentation Pipeline

In this section we will briefly discuss the segmentation pipeline and how it is integrated into the change detection system described in Section 5.3. A more detailed description of each component and discussions about possible alternative choices within each can be seen in [92].

Figure 5.3 shows a high-level graphical overview of how the segmentation system fits within the existing change detection pipeline. Because the number of segments in a particular scene is unknown ahead of time, we perform scene segmentation in an iterative manner. When finding a single segment, the problem is formed as a two-class classification problem where one class represents voxels within the current segment (where the current segment is initiated with some seed voxel) while the other segment represents voxels outside of this segment. This makes use of a general purpose similarity classifier as well as an MRF-based smoothing technique to eliminate outlier segments caused by occasional noise. Each time a segment is identified, the change detection system is used to identify which voxels within it are considered changed, and a final

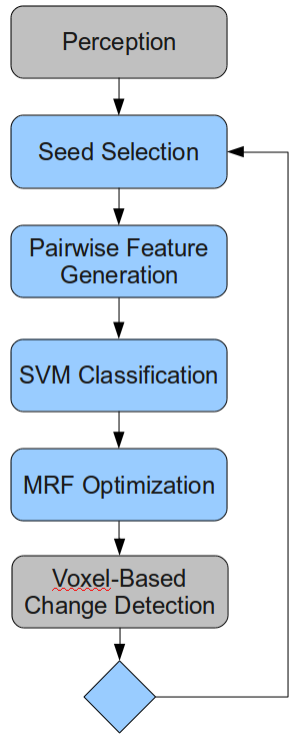


Figure 5.3: The control flow through the segmentation-based change detection system.

decision for the segment as a whole is made after considering all voxel decisions within.

Algorithm 5 presents an overview of the entire pipeline for detecting changes using segmentation and each major step in the segmentation pipeline is briefly described in the following sections. Prior to operation a general purpose classifier is trained as described in Section 5.4.2 to be used through the segmentation process. The scene is then iteratively segmented one segment at a time by finding a suitable seed voxel to start the segment from (Section 5.4.3) and creating and solving an MRF that balances the outputs of the classifier with a neighborhood smoothness constraint (Section 5.4.4).

The procedure is terminated when the scene is fully segmented or a suitable seed voxel cannot be selected. Because segmentation of a scene can change significantly from one iteration to the next due to additional sensor data or randomness within the algorithm, our algorithm relies on the fact that bad segmentations will be rejected by the change detection system until more data is collected and a good segmentation of the relevant area is produced, allowing accurate change detection to be performed.

There are several parameters throughout the system that can significantly effect overall performance. To optimize them we perform a manual coordinate descent optimization on an independent test set, iteratively optimizing one parameter at a time until convergence. The selected values for the most significant parameters are shown in Table 5.1. Because of the system's sensitivity to some of these parameters, this step is critical to achieving good performance (see Figure 5.5 to see the effects of an incorrectly chosen smoothing parameter λ).

Algorithm 5 Segmentation-based change detection algorithm

given: A set of voxel features $(\mathbf{x}_i)_{1...N}$ from the current scene; a sequence of corresponding voxel features from the nearby locations from a previous navigation $S = (\tilde{\mathbf{x}}_i)_{1...M}$; a novelty threshold γ; a segment change threshold η
outputs: A sequence of hypotheses $\mathbf{f} = (f_1(\mathbf{x}_1), f_2(\mathbf{x}_2), \ldots, f_N(\mathbf{x}_N))$ where $f_i(\mathbf{x}_i)$ specifies if voxel $\mathbf{x}_i$ has changed from the previous iteration
while there are any voxels unassigned to a segment **do**
  $s \leftarrow \text{seed}(\mathbf{x})$ {The voxel index selected by the seed selection algorithm (Section 5.4.3)}
  if $s = -1$ **then** {Segmentation termination criteria reached}
    break
  end if
  Run the similarity classifier between x_s and every other unlabeled voxel (Section 5.4.2)
  Create an MRF representing classifier output and neighbor constraints (Section 5.4.4)
  Solve MRF to get $\mathbf{y}$, the labels for all voxels which minimize the MRF energy function in equation 5.10
  $\mathcal{S} \leftarrow \{i \ \ \forall i : y_i = 1\}$ {All voxels labeled as part of current segment}
  $\mathbf{g} \leftarrow \Delta(\mathbf{x}, \mathcal{S}, \gamma)$ {Evaluate each voxel in $\mathcal{S}$ independently using Δ, the original voxel-based change detection system described in Algorithm 4.}
  if $\frac{|\{g_i = 1 \ \ \forall i \in \mathcal{S}\}|}{|\mathbf{g}|} \geq \eta$ **then** {See if percent of segment that has changed passes threshold}
    for each i in $\mathbf{g}$ **do** {Entire segment labeled changed}
      $f_i(x_i) \leftarrow true$
    end for
  else
    for each i in $\mathbf{g}$ **do** {Entire segment labeled unchanged}
      $f_i(x_i) \leftarrow false$
    end for
  end if
end while

Name	Description	Value
k	The number of seed candidates used during seed selection (Section 5.4.3).	12
λ	The MRF smoothing parameter from equation 5.10	0.25
η	The percentage of a segment which must be changed for a changed segment from algorithm 5	45

Table 5.1: A summary of the relevant parameters used for the segmentation-based change detection system. These were optimized using coordinate gradient descent on an independent test set.

An example of a final segmentation of a scene is shown in Figure 5.4(a) while the resulting detected changes with and without the use of this intermediate step are shown in Figures 5.4(b) and 5.4(c) respectively. The result when including the scene segmentation step within the change detection system contains significantly fewer false positives.

5.4.2 Similarity Classifier

To be able to deal with arbitrary environments, we train a single similarity classifier across a variety of known classes to be able to predict how likely any two voxels in a previously unknown environment are to be in the same segment:

$$f(x_i, x_j) = \begin{cases} \geq 0 & \text{if } x_i \text{ and } x_j \text{ are in the same segment} \\ < 0 & \text{otherwise} \end{cases} \tag{5.8}$$

The output of this classifier predicts whether the two voxels are in the same segment or not, and the magnitude of this prediction represents the confidence of the prediction. The key to this classifier is that it operates on any voxels regardless of their classes so that it can generalize well to new environments.

To be able to generalize to any portion of feature space, the classifier utilizes a function ϕ that maps a pair of voxel feature vectors to a vector of features that captures the differences between voxels' raw features: $\phi(x_i, x_j) \rightarrow$ **x'**. Using a scalar function such as Euclidean distance would reduce the problem to that of finding a single difference threshold. By using a higher dimensional difference function allows the classifier to learn more sophisticated non-linear boundaries.

After extensive experimentation, we found that concatenating differences and sums of the original voxels' features worked well since it would allow the classifier to reason about both relative distances in feature space ($|x_i - x_j|$) as well as having some representation of where in the feature space these examples were located ($x_i + x_j$).

Our choice for the classifier balanced computational requirements with performance. After experimentation with a variety of methods, we chose for this classifier to use a kernelized SVM[2] with radial basis functions operating on the output of $\phi(x_i, x_j)$. Using a kernelized SVM rather than a linear SVM increases computation time as a function of the number of stored examples

[2]We used the kernelized SVM implementation provided by libsvm[20].

(approximately 1000 in our case), but allowed the construction of a powerful, non-linear classifier for this task.

Our training examples for this classifier were generated from the labeled set of examples used to develop other parts of the system as described in Section 5.5. We then trained the classifier to accurately classify between members of the same object and those from different categories. We used pairs of randomly selected voxels for this learning task. The sampling was done such that there are an equal number of sampled similar pairs (those which appear in the same log as neighbors and are tagged with the same category label) as non-similar pairs (non-adjacent voxels labeled with different categories). To prevent over-fitting, we perform cross-validation with 20% of the original labeled examples.

Parameter values for the slack variable and kernel bandwidth were then optimized using a grid search at several granularities over possible values across random training and test sets.

This approach allowed us to achieve a 95% overall test accuracy for the classifier.

5.4.3 Seed Voxel Selection

The segmentation system identifies one segment at a time in the environment, starting at a suitable seed voxel and growing that segment to encompass the appropriate portion of the scene. This relies on an efficient and accurate way to identify a seed voxel at each step.

An ideal seed voxel has the property that it is a good representative of all the other voxels that are part of that segment and easily differentiated from other segments' voxels. A poor seed voxel on the other hand might be at the boundary of two segments and have features of both segments without being truly representative of either, making a useful segmentation from that voxel impossible.

To capture these desired properties, we select the seed voxel at each iteration from the set of unsegmented voxels by choosing k of them at random and quantitatively evaluating each candidate's quality. Each potential seed x_i in this randomly chosen set $\mathcal{X}$ is scored using the classifier $c(x_1, x_2)$ trained earlier (Section 5.4.2) as follows:

$$score(x_i) = \sum_{x_j \in \mathcal{X}, i \neq j} \max\{0, c(x_i, x_j)\} \tag{5.9}$$

The seed candidate with the highest score is then chosen as x_s, the seed for the current segmentation iteration. Because $|c(x_i, x_j)|$ is the confidence of the classifier, this metric can be interpreted as choosing the candidate voxel that is most strongly associated with other voxels, meaning it is a good representative of a segment. The voxels that are determined to be different from that voxel (when $c(x_i, x_j) < 0$) are not factored into the metric as they are presumably voxels in other segments.

We found that setting $k = 12$ worked well. A larger k increases the effectiveness of this step but requires more computation since the technique described here is $O(k^2)$.

5.4.4 MRF-Based Segment Identification

As the final step in identifying the segment for each iteration, the problem is represented using a binary MRF that is optimized to account for all classifier evaluations as well as an assumption of local continuity (to prevent tiny segments due to feature noise). This approach is similar to the ground height estimation system in [144] where evidence from noisy laser data is balanced against an assumption of local ground height continuity.

This MRF optimization step is formed as an energy minimization problem where the final solution tries to minimize the degree to which classifier or neighbor constraints are violated. For a segmentation step with N remaining voxels, the final segmentation is represented by the labels $y_{1...N}$ where $y_i = 1$ if node i is in the current segment (same segment as the seed voxel) and $y_i = 0$ otherwise. The first voxel, $\mathbf{x}_1$) is is assumed to be the seed voxel, meaning $y_1 = 1$. The energy optimization problem can therefore be expressed as:

$$\begin{aligned} E(\mathbf{y}_{1...N}) &= \sum_{i=2}^{N} E_c(\mathbf{x}_1, \mathbf{x}_i) + \\ &\quad \lambda \sum_{\{i,j\} \in \mathcal{N}} \delta(y_i = y_j) + \\ &\quad \infty \delta(y_1 \neq 1) \end{aligned} \tag{5.10}$$

$$E_c(\mathbf{x}_1, \mathbf{x}_i) = \left\{ \begin{array}{ll} |c(\mathbf{x}_1, \mathbf{x}_i)| & \text{if } \delta(c(\mathbf{x}_1, \mathbf{x}_i) > 0) \neq y_i \\ 0 & \text{otherwise} \end{array} \right\} \tag{5.11}$$

Where $c(\mathbf{x}_1, \mathbf{x}_i)$ is the output of the similarity classifier described in Section 5.4.2, $\mathcal{N}$ is the set of all neighboring voxels and λ is the weight on the neighbor smoothness constraint. The first term of Equation 5.10 represents the degree to which the segmentation disagrees with the classifier evaluations between the seed voxel and the rest of the voxels. If the assignment of a label does not match the output of the classifier, the labeling is penalized by the magnitude (confidence) of the classifier output. The second term encourages neighborhood continuity by penalizing mismatching neighbors by the smoothness parameter λ. Finally, the third term insures that the seed voxel $\mathbf{x}_1$ is in the current segment.

In general, optimizing an multi-class MRF is NP-hard, but computing the optimal labeling of a binary MRF is a special case that can be solved in polynomial time using the graph-cut algorithm if the energy of the distribution of each pair of binary variables is submodular [63, 148].

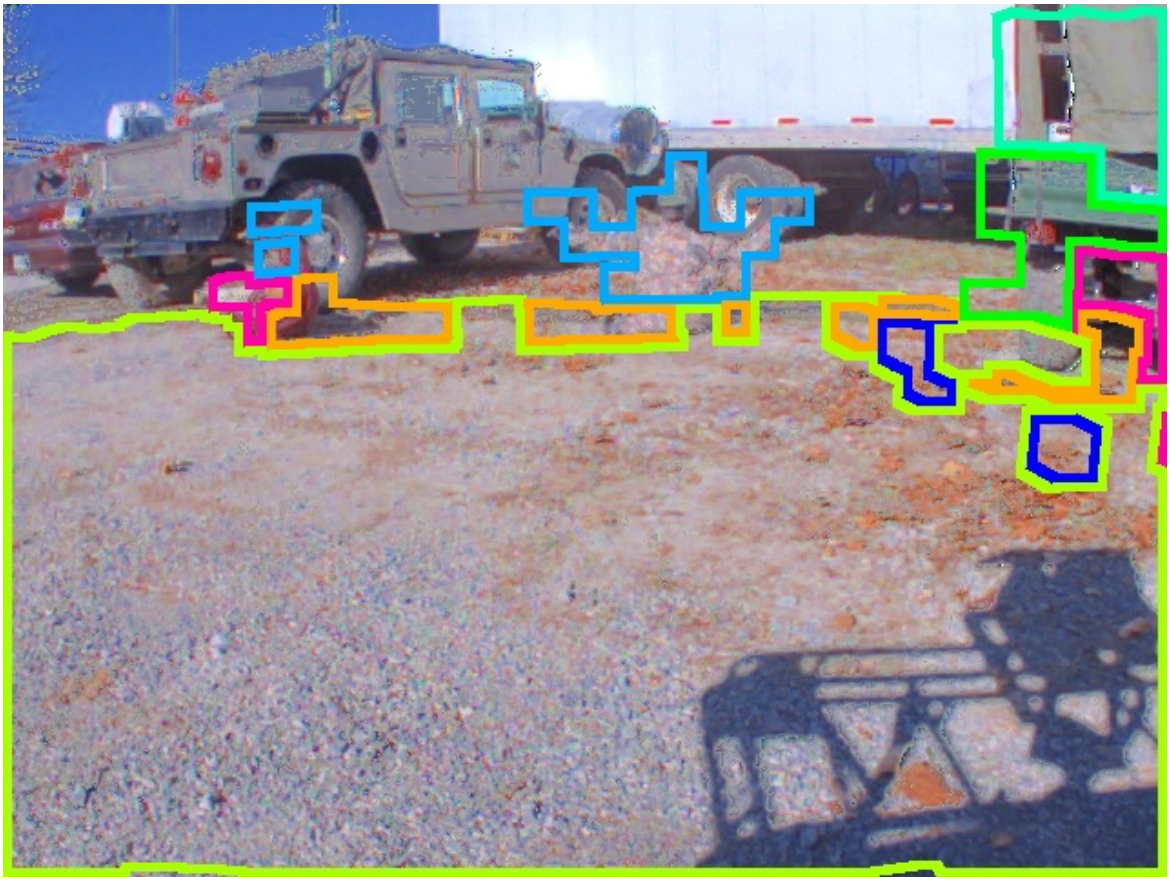
To take advantage of this binary MRF property, we encode the energy optimization from Equation 5.10 into a binary MRF structure as illustrated in Figure 5.6. The initial MRF with neighbor constraints is shown in Figure 5.6(a) (the seed node is identified with a '*'). Figure 5.6(b) shows the MRF with neighbor constraints as well as the output of classifier evaluations between the source node and all other nodes (negative numbers mean the classifier predicts they are not in the same segment).

In Figure 5.6(c) virtual source and sink nodes are created to represent the current segment and all other voxels respectively. All constraints with positive weights are encoded by an edge to the source node while all constraints with negative weights are encoded by an edge to the sink

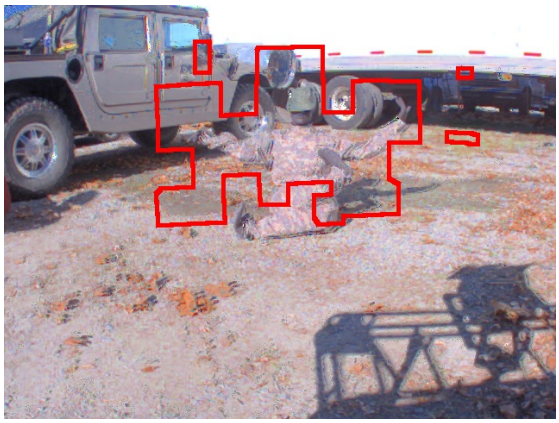
node. All neighbors are tied by edges with weight λ, the neighborhood constraint weight, and an edge of infinite weight ties the source to the seed node ensuring it will be part of the current segment. At this point all edges have positive weights, making each of them submodular and allowing us to solve the MRF optimally and efficiently using graph-cut.

The final outcome for this example problem is shown in Figure 5.6(d) (constraints that are violated by this solution are crossed). Notice how in this example the weak recommendation of the classifier for the top-right node is overruled by the optimal solution due to the neighborhood constraints surrounding it.

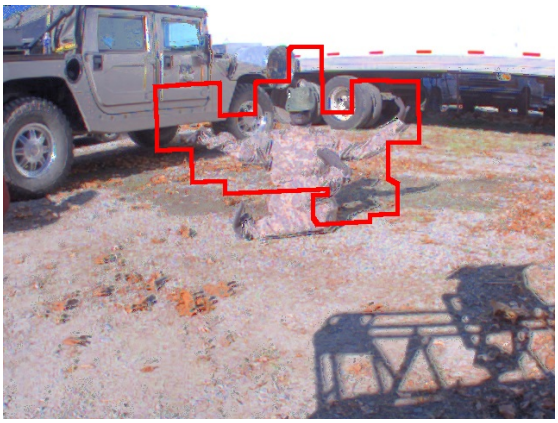
Once the MRF is solved, all voxels that are in the connected set of voxels tied to the source node are then labeled as part of the current segment, completing that iteration of the segmentation system. Those voxels are removed from the original set and the segmentation iterations are repeated on the remaining voxels until the entire scene is segmented or no further acceptable segments can be computed.



(a) Computed segmentation for a scene

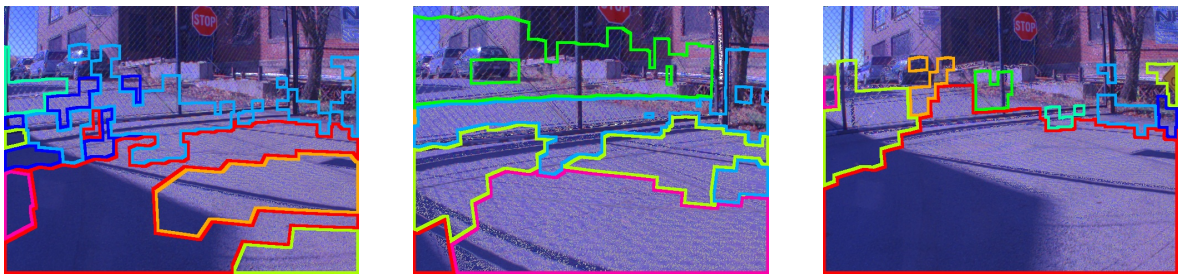


(b) Detected changes without segmentation



(c) Detected changes with segmentation

Figure 5.4: Example output of scene segmentation system and resulting effects on change detection performance in eliminating false positives (manikin was introduced as a change).

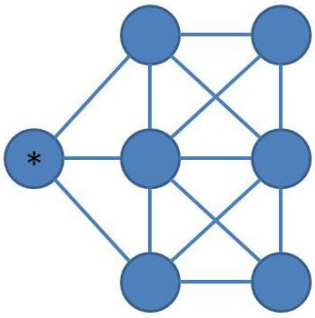


(a) $\lambda = 0.05$: low smoothing parameter results in too many segments, hurting change detection performance

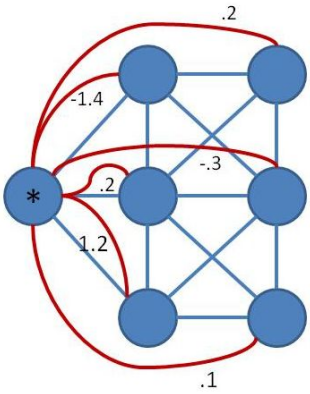
(b) $\lambda = 0.25$: the optimal smoothing parameter results in an adequate segmentation

(c) $\lambda = 5.0$: high smoothing parameter enforces strong segment continuity but does not allow the system to correctly segment the fence which needs to be split between more segments due to feature variation.

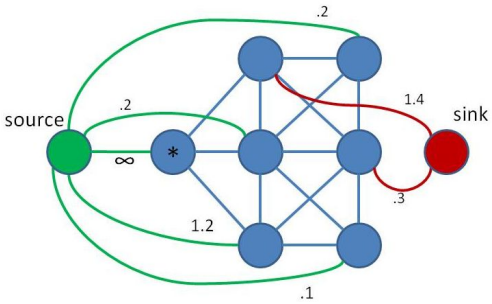
Figure 5.5: Effects of varying smoothness parameter λ.



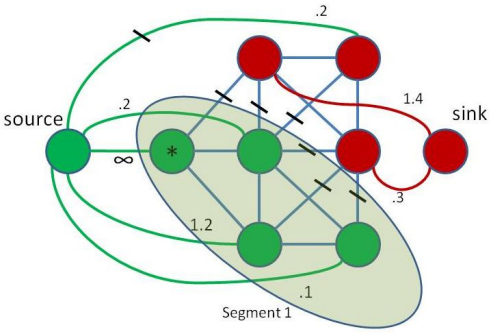
(a) The seed voxel is marked with a '*' and the neighbor constraints are shown in blue.



(b) MRF with neighbor constraints (blue) and classifier constraints between the seed voxel and all other voxels (red).



(c) Graph-cut representation of MRF. Each voxel is tied to a virtual source or sink node depending on the sign of the corresponding classifier evaluation.



(d) Final segmented MRF. Edges with bars through them represent violated constraints.

Figure 5.6: Illustration of the MRF optimization procedure.

5.5 Application to Mobile Robotics

A natural application of these novelty and change detection algorithms is to the field of mobile robots. The Crusher UGV shown in Figure 3.3 was used throughout our novelty detection testing and the smaller E-Gator platform shown in Figure 3.4 was used for all change detection experiments.

As often the case in robotics, we were forced to deal on both systems with a relatively high-dimensional feature space (49 features on Crusher and 30 features on the eGator platform) that contained many features of varying importance and accuracy as described earlier. We dealt with this problem on each platform by computing an MDA-based transformation into a lower dimensional subspace using an available library of hand-labeled examples across many environments as described in Section 5.2.2.

The following experiment on the Crusher platform demonstrates the benefits of this approach for novelty detection. By the same intuition this approach was a key component of the change detection system on the eGator platform, both because of the improved ability to measure similarity as well as by allowing a more compact representation of the perception data from the previous traversal.

Of the available classes, four were used to construct a three-dimensional subspace: road/grass/dirt, rocks, bushes and barrels (see Figure 5.7). A fifth class of examples corresponding to various non-barrel man-made objects (various man-made structures, barriers, vehicles, poles, human-sized plastic figures, etc.) was withheld for testing purposes (see Figure 5.9). It is important to note that class labels are only used initially for generating this transformation and are unnecessary for later processing[3].

After training using the first four classes, examples from those classes as well as the held-out data set of man-made examples were projected onto the first three basis vectors computed by PCA and MDA (see Figure 5.8)[4]. The MDA projections clearly show better separation between the previously-unseen set of man-made examples and the original four classes. As expected, the most overlap occurs with the barrel class as barrels share common properties with other man-made objects such as smooth surfaces, colors, etc. Since we would desire these new examples to be identified as novel relative to the rest of the classes, this separation implies that this is a more suitable subspace for use as a similarity metric within a novelty detection system.

This benefit is clearly visible in the ROC curves shown in Figure 5.10. These curves show the false positive rate vs. the true positive rate for novelty detection under various configurations while varying γ, the threshold for novelty. A random 1500-example subset of each training category was used as the baseline set with respect to which novelty was detected in a held-out test set coming from the same training categories (to detect false-positives) as well as the man-made category (to test for true-positives).

Because our algorithm is optimized for online use, the novelty model can start uninitialized or can be seeded with a sampling of examples used during training so that it can identify areas

[3]For many robust autonomy systems including ours, such data is required regardless for perception system development (for example, for training and validation of onboard classifier systems).

[4]All features were initially rescaled to zero-mean, unit-variance.

that are novel and potentially unsafe to handle with the current perception system.

5.6 Experimental Results

We now report on experimental results for both the novelty detection and change detection algorithms described in the previous chapters. All features were projected into the three-dimensional subspace generated by MDA as described in the previous section. As the environment was explored, perception system features were averaged into $0.8\ cm^2$ grid locations for use as online batches of examples.

5.6.1 Novelty Detection Results

To best exhibit the online novelty detection abilities of our algorithm, we ran experiments with the model initialized to contain no prior examples. Those examples that were identified as novel relative to the current model (composed of everything previously identified as novel) were incorporated into the model as described earlier. Ideal behavior would identify objects as novel initially, while future instances of those objects would no longer be novel. Camera and third-person virtual views (constructed using ladar point clouds colorized by camera imagery) within figures identify novel locations in the scene with a red shading.

Our novelty detection algorithm with query optimization (Algorithm 3) was tested using the Crusher UGV shown in Figure 3.3 in a natural outdoor environment to evaluate its online novelty detection performance (the algorithm ran in real-time on similar hardware using logged data). The test environment traversed by the robot consisted of combinations of road, grass and dirt, a large variety of vegetation, a series of small barrels, several ditches, large heavily-sloped piles of rocks and a long chain-link fence.

The vehicle's initial environment consisted of fairly open terrain with some light vegetation scattered on both sides. As expected, instances of such vegetation were detected as novel the first few times they were seen (see Figure 5.11).

The vehicle then encountered areas of much denser, larger vegetation. Initially, a majority of such vegetation was identified as novel with respect to previous inputs (see Figure 5.12). As the vehicle continued navigating through similar vegetation, the model adapted and no longer identified such stimuli as novel (see Figure 5.13).

Figure 5.14 demonstrates this learning process through a series of overhead images of this initial environment. Of all locations encountered in this run, these images those that are computed to be novel with respect to the vehicle's novelty model as of the shown position. Output is shown at three points in time: near the beginning of navigation, just before initial encounters with dense vegetation and after sensing a small amount of dense vegetation. It is clearly visible how the system adapts quickly as it navigates more of the environment, causing future instances of similar situations to no longer be flagged as novel.

Proceeding through the environment, the vehicle then encounters a series of plastic barrels (see Figure 5.15). As desired, the first several appear as novel with respect to the large variety of

vegetation previously seen while later barrels are no longer novel due to their strong similarity to the initially seen barrels. Similarly, a long stretch of a chain-link fence is identified as novel late in the course (see Figure 3.2). Again, the initial portions of the fence triggered the novelty detection algorithm while later portions were no longer novel due to the algorithm's adaptation to seen data.

The value of such an approach is apparent when considering the UGV perception system's interpretation of the fence compared to more familiar vegetation (see Figure 5.16). Even though the fence is significantly more hazardous to the vehicle than any of the vegetation, it is not in the perception systems experience base and its traversal cost is therefore significantly underestimated. Such a scenario could be catastrophic to a vehicle without such a novelty detection system.

Additional examples of novel instances identified during traversal appear in Figure 5.17.

Overall, the novelty detection algorithm was able to identify all major unique objects (vegetation, barrels, fence, etc.) with a relatively small amount of false positives due to effective adaptation to the environment. When PCA was used to create the feature subspace, errors increased due to the lack of separability between classes. As with any algorithm, the success of this approach is heavily dependent on the quality of features.

Computation time comparisons between Algorithms 2 and 3 on this course highlight the effectiveness of query optimization technique (see Figure 5.18). The average computation time required per novelty query using Algorithm 2 grows with the number of stored examples in a way that would not scale to extremely long operation. Algorithm 3, on the other hand, experiences temporary spikes in average computation time as novel areas are encountered but the query optimization step allows the algorithm to quickly adapt its ordering of stored examples so that it can maintain a bound on computation time throughout navigation and allow effective anytime novelty prediction.

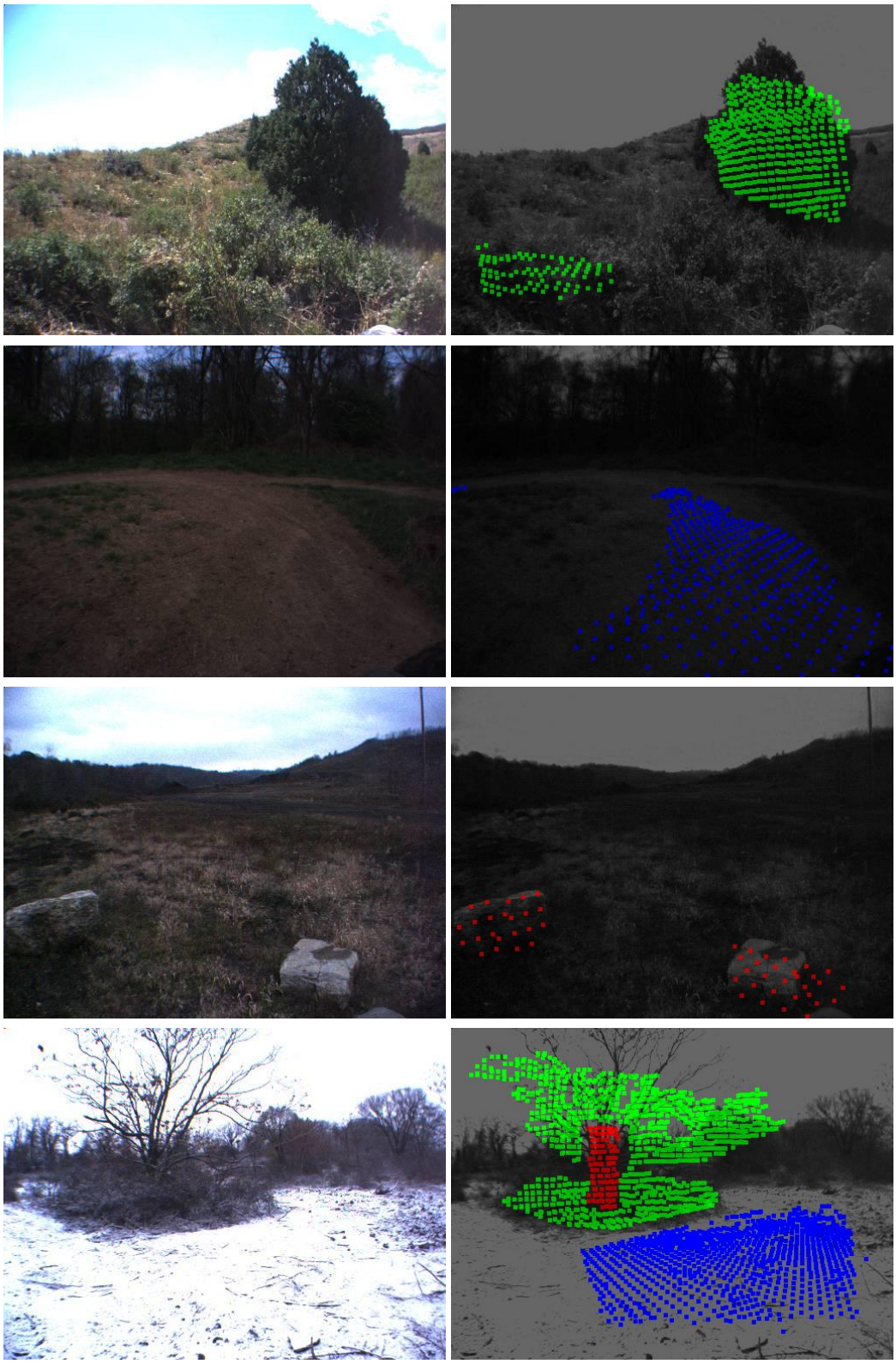


Figure 5.7: Examples of hand labeled class categories (bush, grass, tree trunk, tree branches, dirt, etc.)

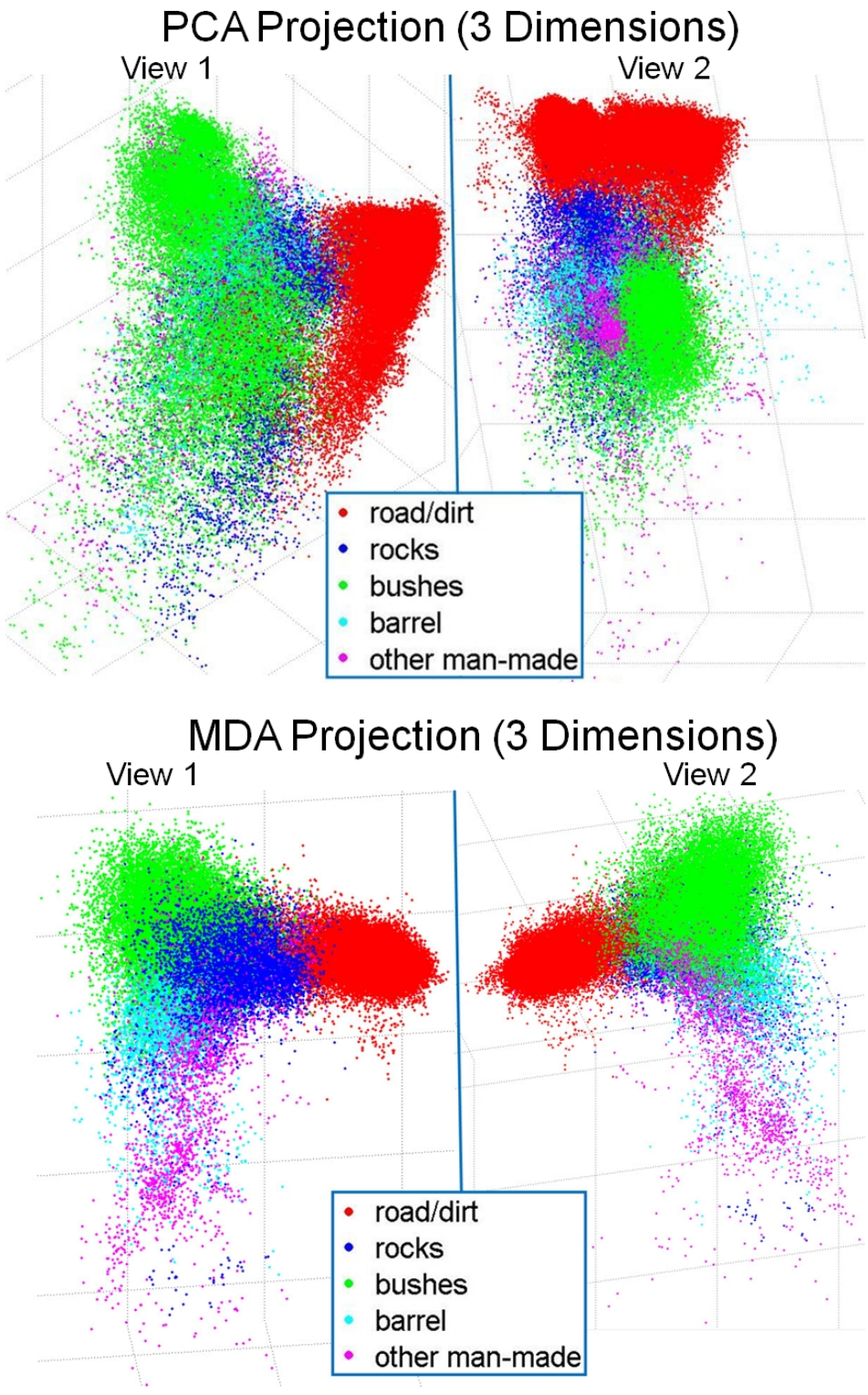


Figure 5.8: All training examples projected onto the subspace defined by the first three basis vectors computed by PCA (top) and MDA (bottom). Only the first four classes were used to construct the subspaces ('other man-made' class was withheld as a test class). The MDA-based projection clearly shows significantly more separation between the new man-made class and the known classes, implying a more suitable subspace for novelty and change detection.

Figure 5.9: Sample labeled examples in the 'other man-made' class used for validation of dimensionality reduction effectiveness. This category excluded instances of barrels which were used as a separate known class.

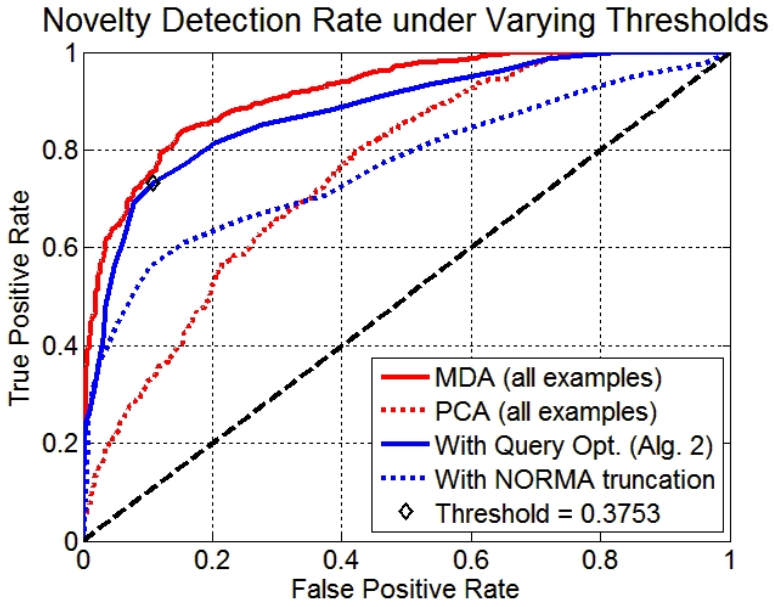


Figure 5.10: ROC analysis of novelty detection performance while varying γ, the threshold for novelty. Performance using lower-dimensionality spaces created through MDA and PCA (shown in Figure 5.8) are shown in solid red and dotted red, respectively. When storage for only 150 examples is available, performances under the query optimization approach described in Algorithm 3 as well as the NORMA truncation approached described in [61] are shown in solid and dotted blue, respectively. For all tests, the learning rate, η, was set to 1, and σ^2 was computed to be 0.9 as described earlier. Novelty detection performance using the MDA-based space was shown to uniformly outperform the PCA-based space and the query optimization approach for dealing with limited memory was shown to uniformly outperform the suggested NORMA approach.

Figure 5.11: Shortly after initialization with no prior novelty model, various small vegetation was detected as novel (identified in red). In several such images throught this section where the marked novel region is hard to see a red arrow has been added to help identify the relevant region.

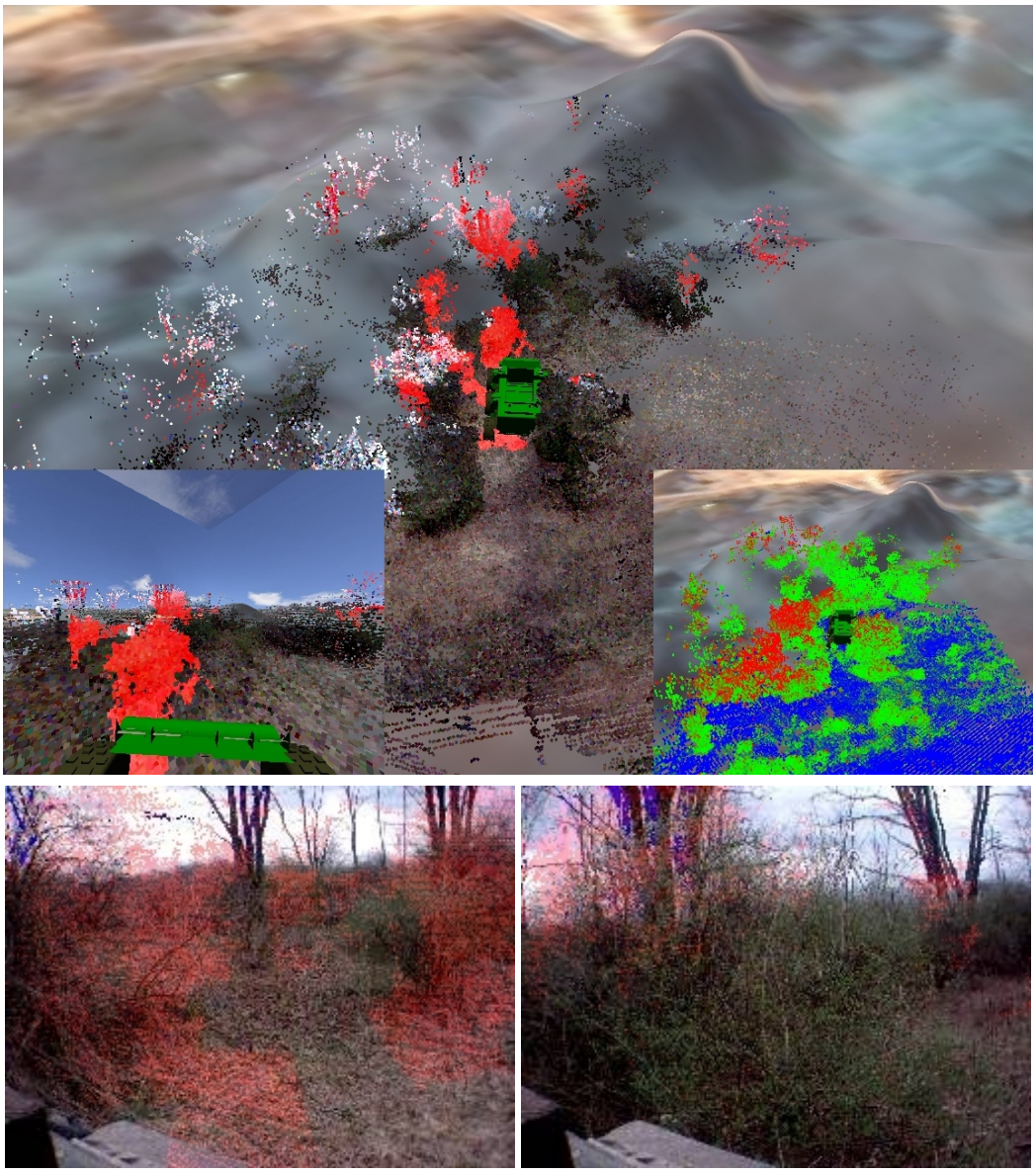


Figure 5.12: A third-person visualization from logged data of the output of the novelty detection system upon an initial encounter with larger and denser vegetation. Because this vegetation is vastly different from anything seen previously, much of it is immediately detected as novel (identified in red). As with all future similar images, insets within the top image show a first-person view (left inset) and the classification of the environment by the perception system into road, vegetation, and solid obstacle in blue, green and red respectively (right inset). The bottom two images show areas of detected novelty visualized on an image from the robot.

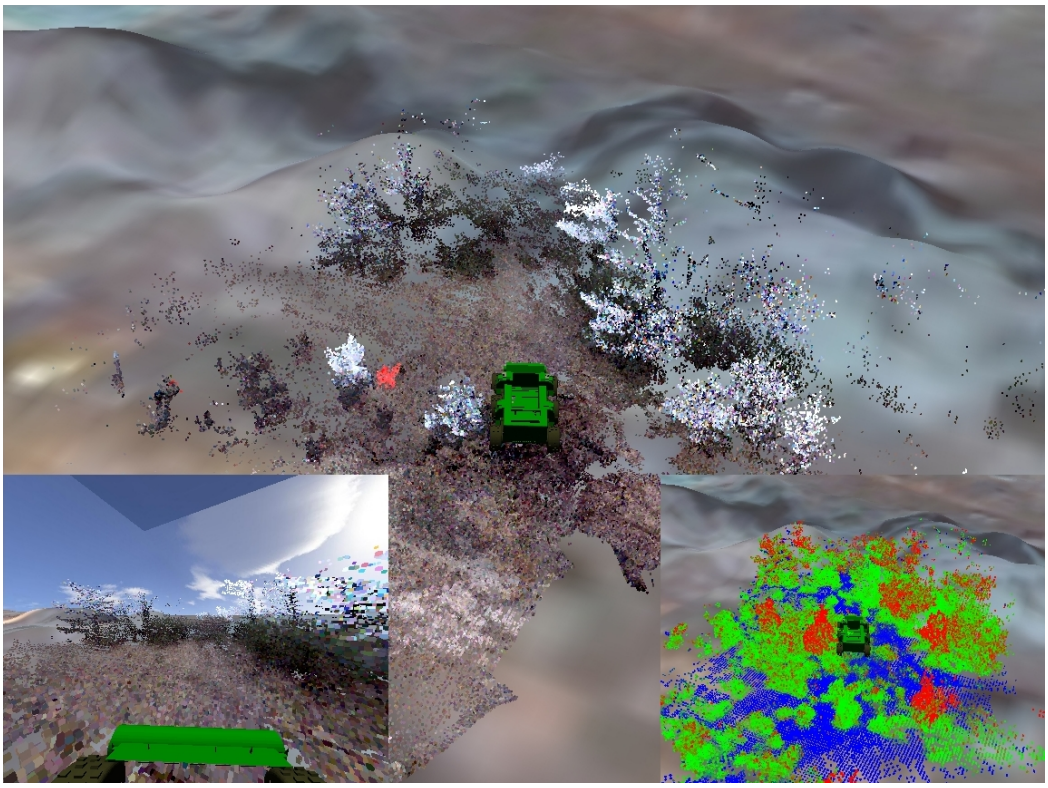


Figure 5.13: Similar vegetation as that shown in Figure 5.12 encountered a short time later. Notice how almost all vegetation is no longer novel due to similarity to previous stimuli.

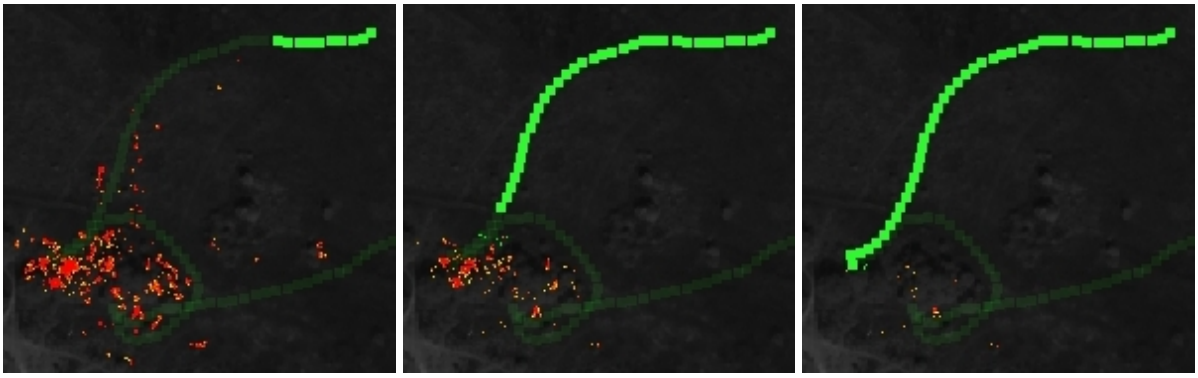


Figure 5.14: Novelty of all future perception input using current novelty model on vegetation-heavy terrain shown in Figures 5.11, 5.12 and 5.13 at three points throughout traversal. Robot's past and future path is shown in light and dark green respectively and novelty of terrain is indicated by a gradient from yellow (moderately novelty) to red (high novelty). Robot is initialized without a prior novelty model.

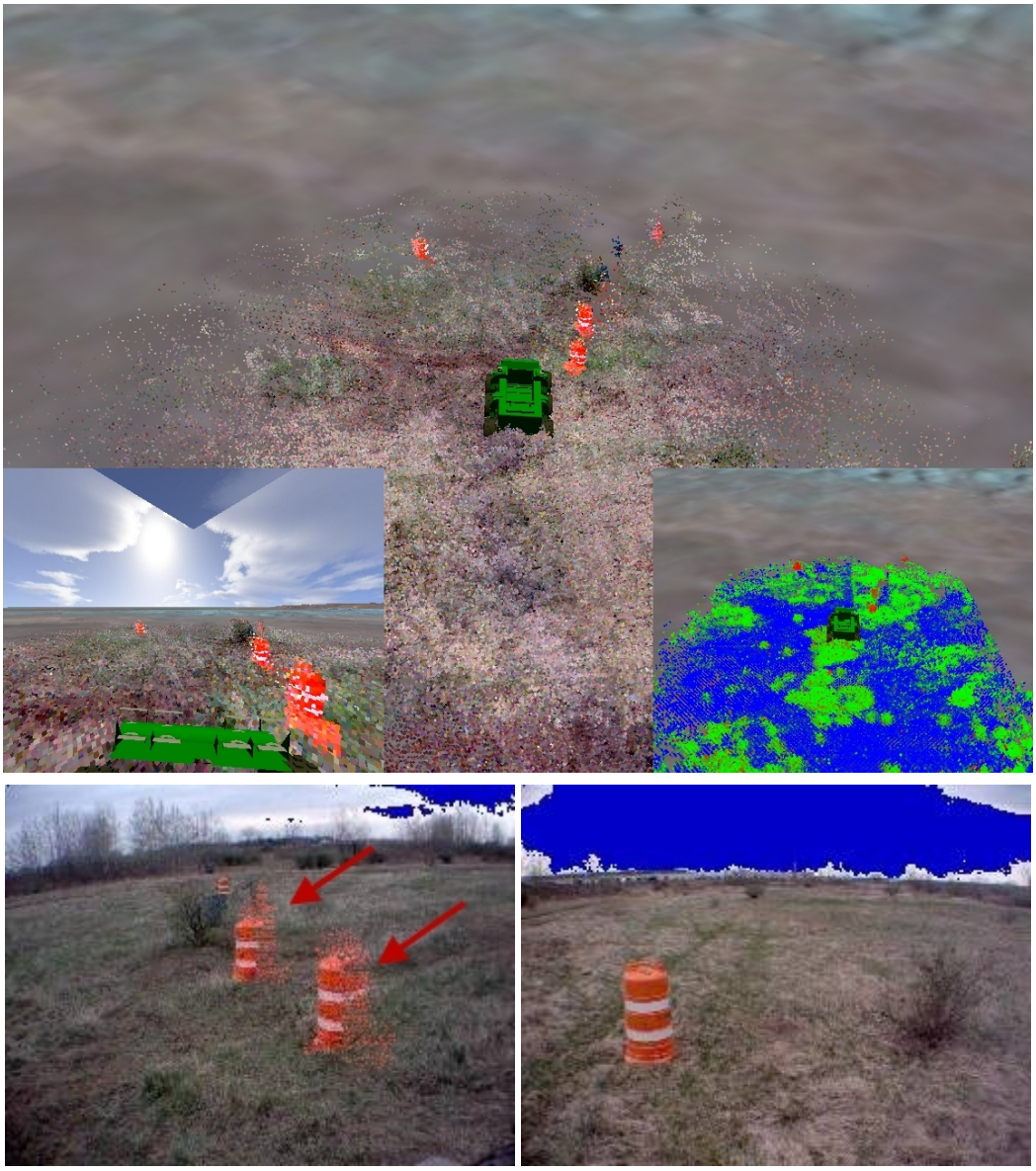


Figure 5.15: Series of barrels encountered later in the course. The initial barrels are detected as novel (red shade) even after significant exposure to a large variety of vegetation (top and left). Later barrels are no longer identified as novel due to online training.

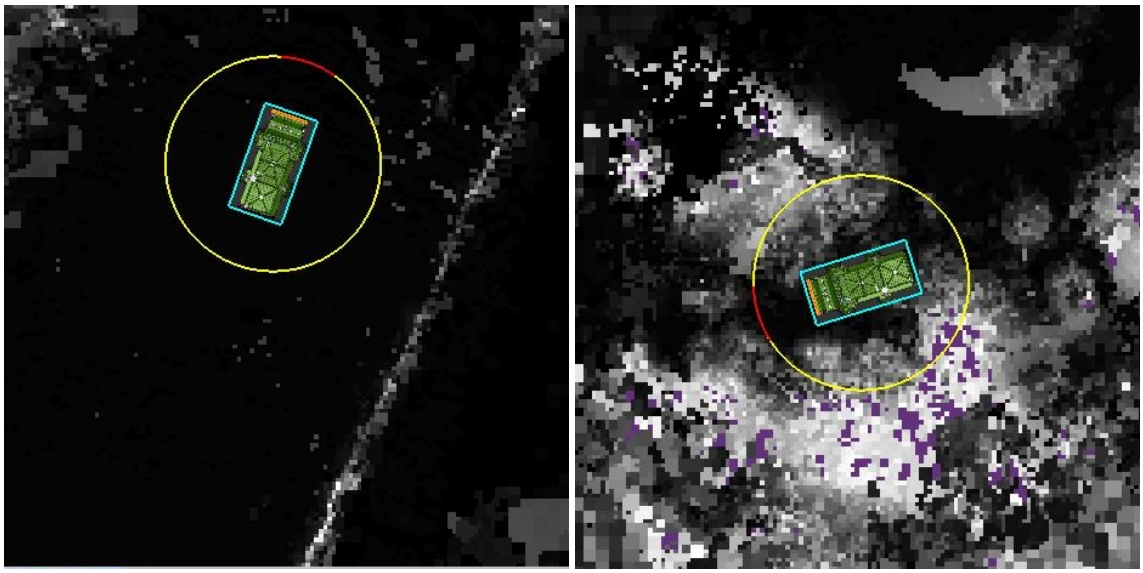


Figure 5.16: The perception system's interpretation of the chain-link fence from Figure 3.2 (left) and the dense bushes from Figure 5.12 (right). Lower and higher traversal cost regions appear in darker and brighter shades of white respectively, with areas that are considered impassable appearing in purple. Even though the fence is significantly more hazardous to the vehicle than any of the vegetation, it is not in the perception system's experience base and its traversal cost is therefore significantly underestimated.

Figure 5.17: Additional examples of novel instances identified during later traversal (red shade): first encounter with a ditch (left) and a large, heavily-sloped pile of rocks (right).

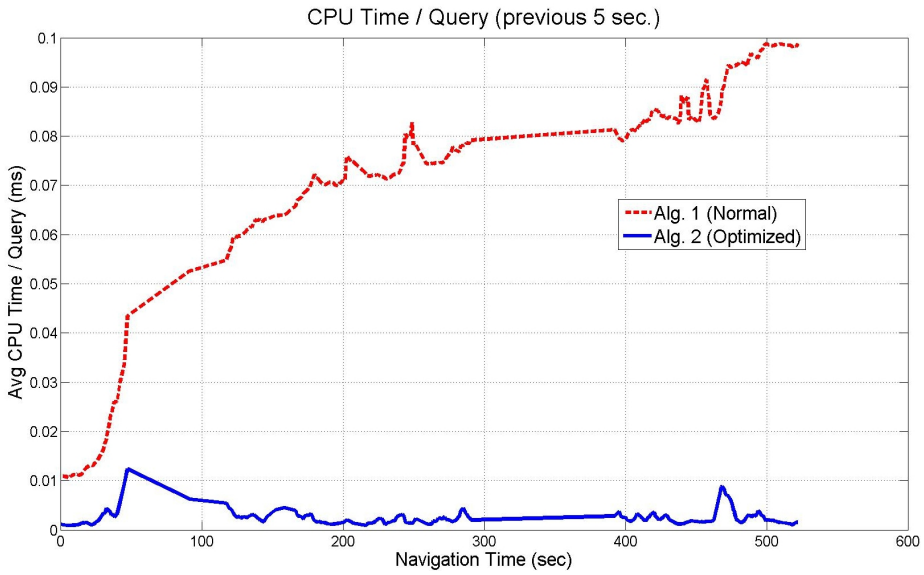


Figure 5.18: Average computation in milliseconds per novelty query on 3.2 GHz CPU for Algorithm 2 (dashed red line) and Algorithm 3 (solid blue line) over the previous 5 seconds throughout navigation. Computational complexity of Algorithm 3 remains bounded due to the order optimization step (line 17). These timings do not include feature computation and projection costs which are identical under both algorithms.

5.6.2 Change Detection Results

The performance of the change detection system was evaluated on 17 pairs of traversals of diverse environments (from logged data) where various changes were introduced for the second traversal. The scenes consisted of a variety of natural and man-made environments and the types of changes tested varied from introductions of manikins (to represent unexpected human presence) to various man-made objects such as barrels, small junk, cones, or cars. These changes were hand-labeled by us to identify the voxels that should have been selected as changed and the performance of the change detection algorithm (both with and without the use of the segmentation system) was evaluated. The system operated on all locations within 12 meters of the robot and within the sensors' fields of view.

Performance was measured for the change detection system with and without the online segmentation component through an ROC analysis of the voxel labeling. By varying γ, the novelty threshold parameter, we were able to change the sensitivity of the change detection system to trade-off between false positives and false negatives and analyze the overall robustness of these systems.

It is also important to note that the ground truth labeling process itself likely added a significant amount of artificial error. In the cluttered 3D environments that we deal with, it was often difficult to accurately label the boundaries of objects. This was particularly true when trying to label objects obscured within vegetation. As a result, even when an object was correctly detected as changed, some error was often still measured during the evaluation of the boundary voxels.

For comparison purposes we also measured performance against several simple occupancy-based approach to the change detection problem. We first considered a direct voxel-based occupancy method where we directly compare each voxel against the same 3D location in the previous traversal's log and consider a location changed if there was a voxel in one scene and not another. Another method we tested against was a region-based occupancy test where a voxel is considered changed only if there is no matching voxel at the same height off of the ground (using the system's estimated ground height) within a one meter window in the previous scene. As expected, such an approach makes fewer false positives (due to a small resistance to registration error) while sacrificing the true positive rate.

In addition to only being able to detect occupancy changes, rather than changes to previously existing objects, these naive approaches are highly suboptimal for several reasons. First, they are extremely vulnerable to registration error throughout navigation. Even small positioning uncertainty will lead to significant false positives, making these approaches impractical for the types of systems we deal with. Further discussion about the problems posed by registration error can be found in Section 5.6.2. Additionally, many objects such as vegetation are not rigid, meaning small variations in wind or season may shift the positions of objects enough to register false changes. Finally, variations will naturally occur due to differences in sensing and driving speeds, causing the perceived sizes of objects to vary across different runs. When coupled with the discretization error resulting from the perception systems voxel-based approach to representing the scene, such naive approaches are destined for poor performance.

We do not quantitatively evaluate the direct performance of the segmentation system itself for several reasons. First, unlike in the case of a classification problem, an ideal segmentation is

highly subjective and makes it impossible to label a ground truth. Also, we use segmentation as an intermediate step in the change detection system pipeline, so for our purposes its performance is better measured by the degree to which it improves the overall system's accuracy.

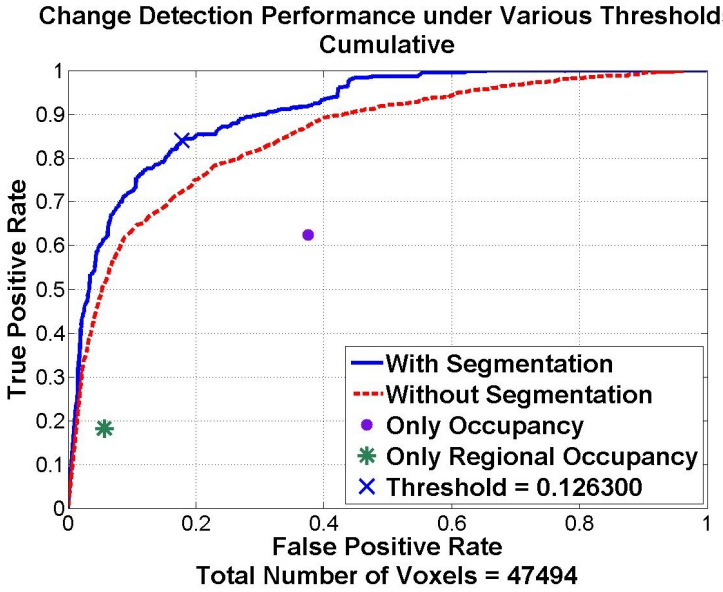


Figure 5.19: Performance of the change detection system when using segmentation (Algorithm 5) and without segmentation (Algorithm 4) across all logs. The optimal value of the threshold parameter γ is marked for the situation where false positives and negatives are penalized equally.

The change detection system that uses scene segmentation identified 84% of the chagned voxels across all all data sets with only a 17.76% false positive rate (see Figure 5.19). To achieve the same true positive rate without the use of segmentation, the system would have to incur a 35% false positive rate. As expected, the performance of the naive occupancy-based algorithm suffered large false-positive rates and significantly underperformed both systems.

The use of online segmentation within the change detection system yielded an improvement in a majority of the scenarios we tested. We also found that the optimal value of the novelty threshold γ (assuming false positives and false negatives are valued equally) remained fairly consistent at approximately 0.1 across all logs. The performance on several individual scenarios, including those where the system struggled, can be seen in Figures 5.4 and 5.20.

It is important to realize that the goal of the scene segmentation system is not to be visually consistent and accurate as is the case for many of the related segmentation approaches discussed. The segmentation results shown throughout these figures often do not precisely fit the objects in the environment for several reasons. First of all, these segmentations are meant to capture consistent regions in feature space rather than objects as interpreted by humans. This feature space includes many features from both color and laser data that varies significantly based on factors like the orientation with respect to the robot, distance from the robot and the lighting in the scene.

Also, due to the role of the segmentation system within the change detection pipeline, it is much preferable to over-segment the scene than under-segment it. Because each segment

is evaluated as a possible change as a whole (the final decision is conservative, only decided on changes when evidence is strong), changed regions in the scene may then be identified as changes through a series of smaller segments that together encompass the object.

Finally, because of the fast pace of online data acquisition, the segmentation for a scene will change constantly, making a poor segmentation for one cycle irrelevant to the overall system's performance (since the regions from that cycle will simply be discarded). As long as each relevant object segmented well in at least one iteration, overall performance will be good.

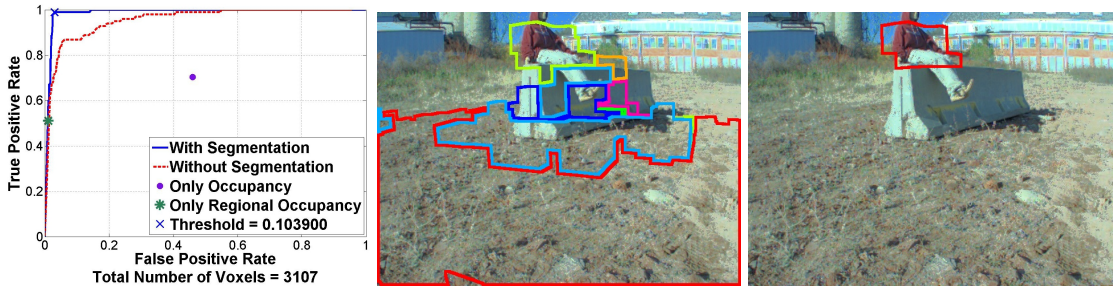
By far the most computationally intensive component of the change detection system is the online segmentation step. We ran the entire autonomy system (including the perception system and feature generation) on logged data at about a fifth of real-time (without segmentation the entire system can easily be run in real-time). We are confident, however, that this system can be made to achieve real-time performance with some optimization effort. The most obvious optimization step would be to parallelize the segmentation process rather than having it process each segment sequentially. Such a parallelization effort could on its own yield the necessary improvements for real-time operation.

Effects of Registration Error

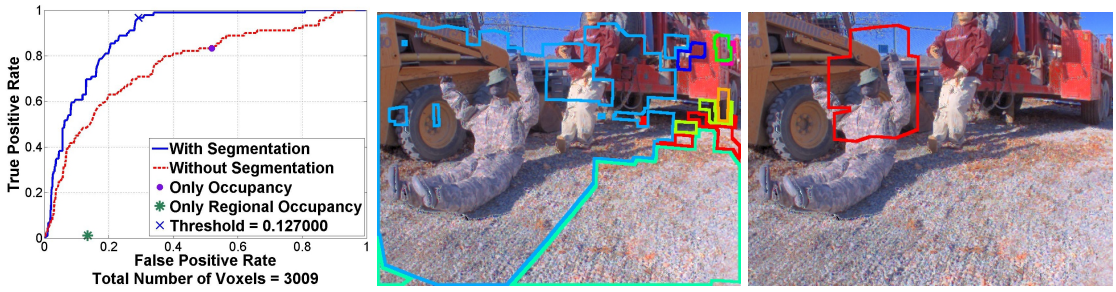
Because the change detection system is comparing the scene against a stored representation of that scene from an earlier time, a key challenge to overcome is dealing with any registration error between one navigation and another. As mentioned previously, GPS systems often produce especially large vertical (z-axis) errors. This is clearly visible in the visualization in Figure 5.21 of perception data from two traversals of the same scene at different times where a vertical drift of almost 2 meters is present. We correct for this type of error by using our online ground plane estimation system as a baseline for the vertical axis rather than the GPS measurement.

Dealing with positioning (x-axis and y-axis) error is a more challenging problem. Unless the system uses extremely high-end positioning tools, registration errors of up to a meter or more are common. To show how our system responds to more severe registration error, we artificially mis-registered each pair of logs with added errors of varying amounts (this is in addition to the inherent error already present in the log pairs). To account for the higher expected registration noise, the radius from the previous navigation from which voxels are used for comparison are grown accordingly. Figure 5.22 shows how our system performs under these conditions.

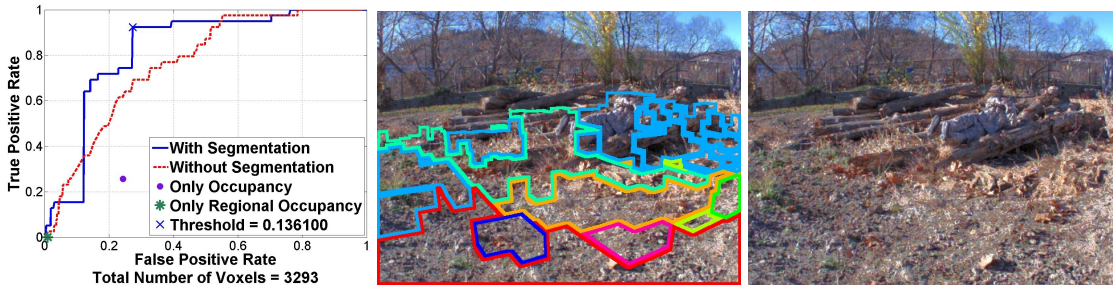
As shown in this figure, the described change detection algorithm performs well under these increasingly difficult conditions. The system's ability to not rely on strict position matches between voxels allows it to keep a relatively low rate of false positives. The segmentation system further helps by eliminating local error caused by portions of objects no longer aligning. Because decisions are made on contiguous segments as a whole, the system can withstand some amount of mis-registration for portions of these segments while limiting the amount of additional false positives. This improved resistance to registration error can clearly be seen in Figure 5.23 which shows the area under each ROC curve from Figure 5.22. The area-under-curve measure is a common way of evaluating the robustness of an algorithm under varying thresholds.



(a) Despite never having trained on an object this shape or color, the system has no problem identifying the manikin as a change to the scene



(b) The first manikin was successfully detected but the later one was missed due to imperfect segmentation.



(c) In this exceptionally difficult scenario, the segmentation system was unable to distinguish camouflaged manikin from the logs resulting in a high false-positive rate before any correct changes were detected. It is possible that in this scenario the features generated for the environment are simply not rich enough to adequately characterize this scene.

(d) Two manikins introduced into the scene were detected perfectly. The second was pinned underneath the vehicle, making accurate detection of this manikin especially difficult.

Figure 5.20: Performance on selected individual scenarios. The ROC performance, scene segmentation, and detected changes are shown from left to right. The occupancy tests in the ROC plots refer to the tests described in Section 5.6.2.

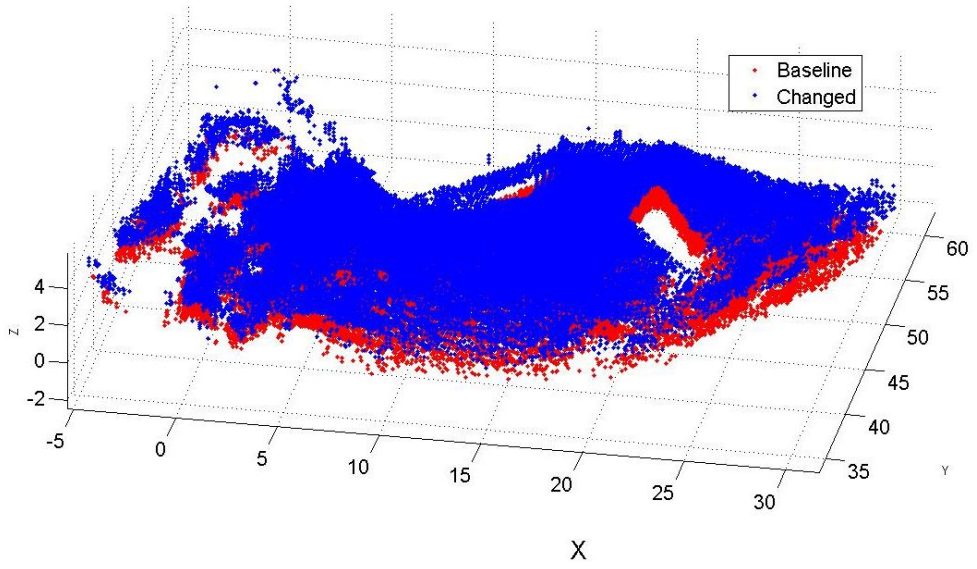


Figure 5.21: Visualization of perception data from two navigations of a scene at different times. While positional error is often relatively small, vertical error is usually much larger (often several meters).

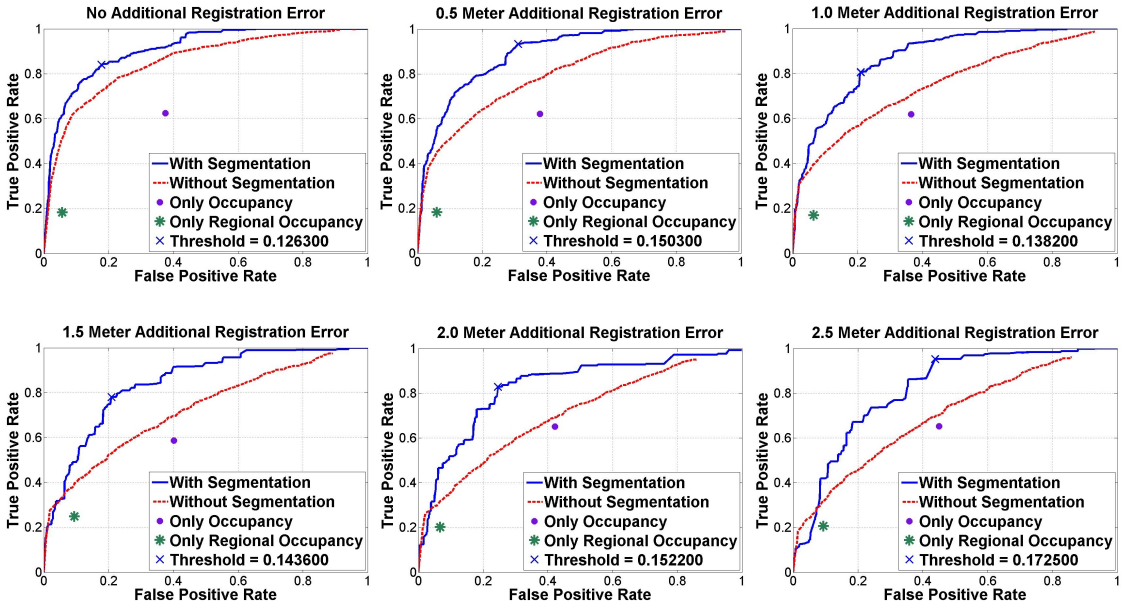


Figure 5.22: Change in performance as additional registration error between past and present navigation is added (in addition to the already existing error inherent in the logs). The change detection system with segmentation is most resilient to this added noise while the other methods begin to suffer.

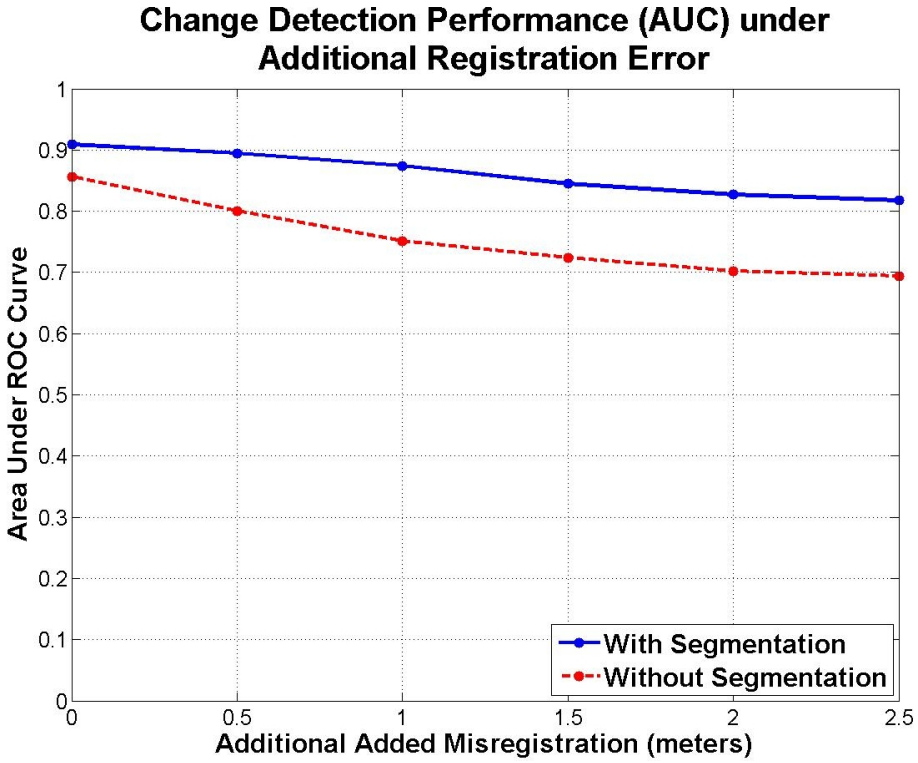


Figure 5.23: Area-under-curve for the ROC curves of each approach under additional introduced registration error. Both systems maintain their performance well even under large additional registration error, but the segmentation-based change detection system is clearly more resistant due to its improved ability to eliminate false positives.

Chapter 6

Online Candidate Selection

As we argued previously, many UGVs possess competent autonomy systems that are capable of dealing with a wide variety of situations. These systems are not perfect, however, and their performance degrades as characteristics of their environment begin to diverge from the environments used throughout training. In some domains it is therefore reasonable to assume that a human operator may be available for short periods of time to provide remote supervision or teleoperation. The responsibility of deciding how and when to use this remote operator assistance to improve performance and mitigate risk lies with the autonomy system.

The final portion of this thesis is devoted to a candidate selection system that observes the performance of the autonomous vehicle in particular situations and compares that performance to remote human-control in similar situations [116]. When the vehicle encounters such situations in the future, it will be able to make a decision about which candidate will perform better.

It is important for such a system to be well-suited for online use. Not only is it unpredictable in advance how well the autonomy system will perform in novel situations, but human operator performance can also vary depending on factors such as bandwidth limitations, operator handicaps such as limited skill or familiarity with the interface, fatigue and weather conditions. When little prior knowledge about the operators' abilities is available, a learning system can observe the performance of the autonomous vehicle in particular situations and compare that to performance under remote human-control in similar situations. When the vehicle encounters similar situations in the future, it can then invoke whichever expert demonstrated better performance: the remote human or autonomous vehicle. Such a capability would enable a single operator to assist many UGVs, ensuring peak performance for the entire team with minimal human involvement.

We pursue this problem using an on-line, reinforcement learning approach and demonstrate its performance on logged data from the Crusher vehicle shown in Figure 3.3. The candidate selection system's goal is to learn to interpret available overhead sensor data in order to make decisions that maximize its overall long-term performance. This inevitably becomes a trade-off between exploring candidates' performance in situations that will allow it to learn more about the world and taking advantage of their learned models to maximize current performance.

One way to take advantage of such a system is to combine limited human availability with the online novelty detection capabilities of Section 5. Since it is impossible to predict what a

UGV may encounter, the key to success is for the UGV to seek help *before* it experiences a major failure. The human can then either inform the robot that it is safe to proceed or handle the novel situation himself, significantly reducing mission risk. Once the significant mission risk posed by novel situations is mitigated, the candidate selection system could be utilized to further optimize the systems navigational performance through selective human control. We call such a two-pronged approach the *Assisted-Autonomy Framework*. We explore the novelty detection and candidate selection problems independently in this thesis, although it is likely that a combined approach may be most effective for certain situations.

Because this technique is applicable to any scenario with repeated online choices, we also show how this approach can be used to deal with scenarios where limited high-resolution overhead data is available to aid the robot in navigating through an environment. The Digital Terran Elevation Data (DTED) level of an overhead elevation data set specifies its density of coverage. Higher resolution overhead data can be used to produce more accurate traversal cost estimates that the UGV can use for better prior path computation but often require expensive and time-consuming aerial surveying and a large amount of bandwidth if remotely supplied to the vehicle. In scenarios where the availability of such data is limited, our algorithm can be extended to allow the robot learn to identify the situations where it will most benefit from high resolution data in order to allocate it to areas that maximize its impact.

The next section presents background on adjustable autonomy techniques and some example applications. Section 6.2 presents our online candidate selection algorithm, followed by experimental results in Section 6.3.

6.1 Related Work

As robotic systems continue to play a larger role in our societies, there has been increased attention on how to optimize the interactions between humans and robots. This field is often referred to as Human-Robot Interaction, or HRI.

Many researchers have investigated approaches for heterogeneous human-robot teams. In such a scenario, humans and robots act as independent agents that must cooperate on a given task. Such techniques have been explored in domains such as the "Treasure Hunt" scenario [27, 56], robot soccer [4], forest fire monitoring [18], border patrol [39] and search and rescue assistance [19, 94, 104, 147].

We deal instead with the scenario where a human can contribute limited attention to improve a robotic system's performance but is not himself an independent agent in the scenario. In this case, a robotic system operates somewhere on the spectrum between full autonomy, where there is no human involvement, and full tele-operation, where the human is in complete control at all times. Scenarios where the degree and methods of human interactions with robots within a system can be varied dynamically in order to optimize performance are often referred to as ones of *sliding autonomy* or *adjustable autonomy*. While most mobile robot systems tend to lie on one of the two extremes of this spectrum, effectively balancing autonomy with limited human involvement can lead to significant improvements in safety, efficiency and overall cost.

The extreme of full tele-operation is already common in many tasks such as remote bomb dis-

posal or reconnaissance [146], operation in hazardous environments [23, 24] and robotic surgery [126]. On the other extreme, full autonomy has been heavily studying in the research community but often is unable to transition into real-world applications due to high reliability requirements and cost constraints. We argue that the way to optimizing the value and impact of robotic systems is to find a compromise on this spectrum that balances cost and development time with autonomous abilities, allowing these systems to be fielded years before otherwise possible.

In some scenarios where the human is the primary operator, the autonomy system is intended to aid by request or when it detects a dangerous situation. This is especially common in various driving assistance systems that are gradually becoming available in high-end vehicles. These include systems that detect drowsiness [43], provide parking assistance [139], automate cruise control [136], and provide situation aware brake assistance for collision avoidance [83]. Similar techniques have also been applied to trains, busses, semi-trucks, many forms of public transit [10], and aiding flight and air traffic control [131].

Similar approaches have been applied to surgery to decrease surgeon fatigue and improve performance. Through the use of force feedback and vision systems, researchers are developing a hybrid control scheme to perform basic subtasks in robotic-assisted laparoscopic surgery [66]. Such systems can automate repetitive tasks such as the cleaning-suction process, a simple yet tiring procedure that often limits durations of surgical procedures.

Much of the success in space exploration has been largely due to robotic technologies. The immense cost of rovers that are to operate on Mars or the moon makes it imperative that their safety is protected. While the space program has historically been conservative in introducing autonomous capabilities to robotic systems, NASA and the research community are gradually beginning to explore the potential benefits of limited autonomous operation. While tele-operation is possible in most cases, the distances over which commands must travel make the associated time delays troublesome. Commands sent to the moon would experience up to a 2.5 second delay while tele-operation of a rover on the surface of Mars can experience delays of up to 45 minutes.

A safeguarded tele-operation approach sharing control of the rover using a command fusion strategy was proposed for time-delayed remote driving [65]. In benign situations, users remotely drive the rover, while in hazardous situations, a safeguarding system running on-board the rover overwrites user commands to ensure vehicle safety and deal with the user's inability to evade obstacles effectively due to the time delay.

Specifying series of waypoints at once can partially alleviate this issue but would be still hindered by the operator's limited field of view at any given point in time. In an effort to increase performance through autonomy, NASA began utilizing navigational autonomy at times on its Mars Exploration Rovers, Spirit and Opportunity. In May 2005, NASA integrated a version of the Field D* algorithm into the navigation software of the rovers, enabling local and global planning [17, 33]. Such approaches have the potential to improve productivity while significantly reducing supervision requirements and personnel costs. Others are exploring the roles of adjustable autonomy in future space missions to allow humans to closely interact with robotic systems at whatever level of control is most appropriate [28].

The formulation for most of the situations described above is sometimes referred to as *user-based autonomy*: adjustable autonomy is driven by the need to support user control [76]. We

instead deal with scenarios within the *agent-based autonomy* formulation where autonomy is the default mode of operation and an agent explicitly reasons about whether and when to transfer decision-making control to a human. Since interrupting the human often has high costs, complexity falls on defining an acceptable *transfer-of-control* strategy.

The Trestle system, for example, consists of three different robots that must work together to assemble a small structure from individual beams [46, 106]. These robots can either function autonomously or through tele-operation from a human. Through decision trees and Markov Decision Processes trained from prior performance data, the system was able to utilize the human intermittently to achieve a balance between human use and overall performance. However, in scenarios such as the one we address, prior performance information is often unavailable and the system must learn such models online. Furthermore, this system is able to request human assistance to correct failure, a luxury that is not available in many situations.

For single-agent systems some have suggested relinquishing control when there is an expectation of high benefit [47, 49] or the degree of uncertainty is high [44]. In scenarios where a single human must supervise multiple robots, adjustable autonomy becomes a requirement. Scerri et al. have extensively studied transfer-of-control strategies for large multi-agent teams using Markov Decision Processes [101]. Similar techniques have enabled NASA to replace a full-time multi-person operating staff for supervising satellite behavior with automated systems that signal for human help only when unexpected events occur [14].

Fong introduced the idea of *collaborative control* to allow the human and the robot to engage in dialog to exchange information, ask questions or to resolve differences, allowing a single human to supervise multiple robots simultaneously [34, 35]. This is accomplished through a set of approximately thirty defined queries enabled at specific situations. While this improved robot performance in situations that matched the pre-defined system specifications, the system would not extend well to novel situations due to its rigid definition and lack of online learning abilities.

Goodrich and Schultz have written an extensive survey article on the field of Human-Robot Interaction exploring many additional approaches and applications [42].

The key difference in our approach from the above-mentioned approaches is that we do not constrain the system by any pre-determined rules or models. Since it is not possible to prepare for all possible circumstances a robot may encounter, the ability to learn online the capabilities of each potential expert allows our systems to better adapt to more diverse and challenging environments.

6.2 Approach

6.2.1 Contextual Multi-Armed Bandit Setting

The candidate selection problem involves choosing an operator for each encountered situation from a set of candidate systems, in our case the autonomy system and the human tele-operator, whose performance we assume comes from some unknown distribution. It is therefore intuitive to frame this problem as an instance of the commonly studied *multi-armed bandit* problem [6, 69, 97]. Bandit problems are relevant to a wide range of domains such as statistics, economics

and clinical trial decisions [38, 68, 142].

In the k-armed bandit setting, at each time step the world chooses k losses (or rewards), $l_1, \ldots, l_k$, and the player makes a choice of an arm $i \in \{1, k\}$ without knowledge of the hidden losses. The player then observers only the loss l_i corresponding to the chosen arm. Since the loss distributions are unknown, there is an inevitable conflict between minimizing the immediate loss and gathering information that will be useful for long-term performance. This is often referred to as the *exploration-exploitation trade-off* since we must choose between *exploring* our unknown loss distributions and *exploiting* the arm we currently believe to be best.

We deal with a more suitable variation of this setting called the *contextual bandits* setting where at each time step t the player also observes some contextual information x_t which can be used to determine which arm to pull [71, 140]. We compute these features from commonly available overhead imagery and DTED 3 elevation data for the given environment as described in Section 3.3.2 and convolve them with a Gaussian kernel in order to blur the data, in effect introducing an influence from surrounding areas into each location. This creates a more realistic modeling problem since navigational performance at a given location is heavily influenced by factors from the surrounding area.

As is common with bandit problems, our goal is to minimize regret, the difference between the performance of the algorithm and that of the optimal algorithm in hindsight:

$$R = \sum_{t=1}^{T} (l_t - l_t^*) \tag{6.1}$$

where l_t^* is the loss incurred in round t by the optimal strategy.

6.2.2 Exploration-Exploitation Trade-off

Finding the right balance between exploration and exploitation when dealing with a bandit setting is one of the core problems in the field. For the standard multi-armed bandit problem, some simple approaches include:

ϵ-greedy strategy. The best known arm is selected for a proportion $1 - \epsilon$ of the time and a random arm is selected for a proportion ϵ [141].

ϵ-first strategy. A pure exploration phase is followed by a pure exploitation phase. For an experiment of length T, the exploration phase where a random arm is chosen occupies ϵT steps and the exploitation phase where the best arm is chosen occupies the remaining $(1 - \epsilon)T$ steps.

ϵ-decreasing strategy. Similar to the ϵ-greedy strategy, except that the value of ϵ decreases as the experiment progresses, resulting in higher exploration earlier in the experiment and more exploitative behavior later.

Approaches such as these are not well-suited our problem since they ignore the availability of contextual information. We therefore choose to deal with the exploration-exploitation trade-off through the use of confidence bounds. With a model that is able to supply confidence bounds, the widths of the confidence bounds reflect the uncertainty of the algorithm's knowledge. By

choosing the candidate with the highest upper confidence bound at each time step, the algorithm elegantly trades off between exploration and exploitation. When uncertainty is high, choosing that candidate will provide information that will quickly reduce uncertainty in that region of the model. As we gain knowledge about each candidate, confidence bounds will shrink and we will choose the candidate with the highest expected performance. This approach was well-justified for the bandits setting and shown to have small regret [5].

6.2.3 Linear Optimization as Multi-Armed Bandits Problem

An algorithm that was very influential on our approach was a linear optimization analog of the k-armed bandits problem proposed by Dani et al. where rather than finitely many arms, the decision set is a compact subset $D \subset \mathbb{R}^n$ [26]. At each step, the algorithm must choose a decision $x_t \in D$, and each choice results in a loss $l_t = c_t(x_t)$ where c_t is assumed to be a fixed linear function with some amount of additional noise.

Their algorithm utilizes upper confidence bounds by maintaining an ellipsoidal region in which the optimal decision μ is contained with high probability. Suppose decisions $x_1, \ldots, x_{t-1}$ have been made, incurring corresponding losses $l_1, \ldots, l_{t-1}$. Then their estimate $\hat{\mu}$ to the true cost vector μ can be constructed by minimizing the square loss:

$$\hat{\mu} = \underset{v}{\operatorname{argmin}}\ \mathcal{L}(v), \text{ where } \mathcal{L}(v) = \sum_{\tau<t}(v^T x_\tau - l_\tau)^2 \tag{6.2}$$

A natural confidence region for μ is then the set v of decisions for which $\mathcal{L}(v)$ exceeds $\mathcal{L}(\hat{\mu})$ by at most some amount β:

$$\{v|\mathcal{L}(v) - \mathcal{L}(\hat{\mu}_t) <= \beta\} \tag{6.3}$$

The confidence region at time t, B_t, is defined to be the ellipsoid that contains the region defined in (6.3). The decision at the next round is then the greedy optimistic decision:

$$x_t = \underset{x \in D}{\operatorname{argmin}}\ \min_{v \in B_t}(v^T x) \tag{6.4}$$

We propose to utilize a variation of this approach as described in the following section.

6.2.4 Formalization

We frame online candidate selection problems as follows. At each time step t, we get some contextual features x_t for our environment and must choose from one of k candidates to operate the robot for that time step[1]. These features will be generated from either on-board or overhead sources using similar methods to those described earlier and will contain information over a

[1] In the case of choosing between a human and the autonomy system, $k = 2$. We discuss this problem in the more general case as it could also be applied to choosing between multiple autonomy systems, multiple human operators, etc.

broader area. The goal in such a setting is to minimize the amount of time spent dealing with the situation at that time step, measured by the period of time it takes the robot to enter and exit a 3 meter radius window around that location.

After each selection, the algorithm observes the noisy feedback l_t^i of only the chosen candidate i. We model the distribution for l_t^i as a Gaussian whose mean is a linear function of the contextual features x_t:

$$E(l_t^i|\mu^i, x_t) = \mu^i x_t \tag{6.5}$$

We assume the estimates have Gaussian noise and are therefore distributed:

$$\tilde{l_t^i} \sim Normal(l_t^i, \sigma^2) \tag{6.6}$$

Our problem differs from the linear optimization scenario of Dani et al. described above in that we do not choose $x_t \in D$ at each time step but rather receive a fixed x_t and must choose among our k candidates. We are therefore tracking k instances of the linear optimization problem in parallel, one for each candidate. This makes our confidence region problem simpler as we only have k alternatives to evaluate, the upper confidence estimate of x_t for each of the k candidates, rather than all hypotheses contained in the ellipsoid B_t. Bayesian Linear Regression as described in Section 4.2.1 is therefore an appropriate algorithm for maintaining estimates for each μ^i and generating upper confidence-based predictions.

6.3 Experimental Results

We validate this candidate selection algorithm offline through the following two applications relevant to mobile robot navigation.

6.3.1 Adjustable Autonomy

While we do not have the system infrastructure to be able to trade-off online between tele-operation and autonomous vehicle control, we simulated such an online scenario by using a pair of logged traversals of the same long course in western Pennsylvania by each candidate: a human tele-operator using a high-bandwidth camera system and the autonomy system. All locations where the path of the human driver and the autonomous driver were in sufficient proximity were used as a test point for the system. As the algorithm chose a candidate, the traversal time for only the specified candidate was revealed to the algorithm.

The course and estimated relative performance of each candidate using a trained model appear in Figure 6.1. Quantitative results comparing our algorithm to various alternatives appear in Figure 6.2 and Table 6.1.

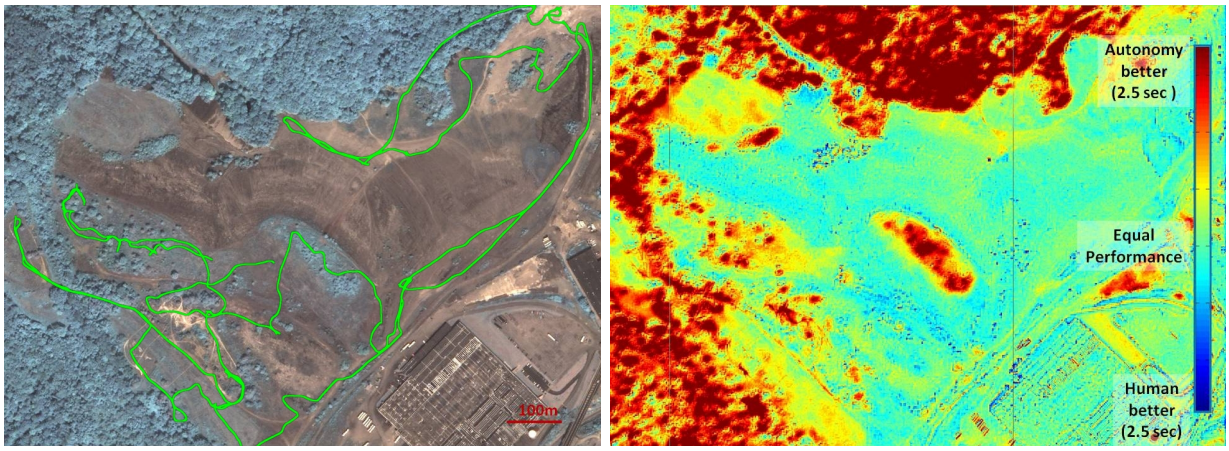


Figure 6.1: Aerial image of test site with course driven using each operating mode (left) and the estimated differences in traversal time in seconds per meter for this site using the final models learned by the online candidate selection algorithm (right). The algorithm found that human performance tended to excel in open areas where the human was better able to interpret sparse obstacles and drive aggressively and at perimeters of heavy obstacles when the human's situational awareness allowed him to better interact with the environment.

Table 6.1: Online Operator Selection Performance

Algorithm	Cumulative Time (seconds)[a]	Percent Improvement over Always-Human
Online Algorithm	9551.7	9.41
Optimal	7809.3	25.94
Worst-Case	12791.0	-21.31
Always-Human	10544.4	0.00
Always-Autonomy	10055.9	4.63
Random Driver	10307.4	2.95

[a] Note that since 3 meter regions at example locations often overlapped with each other, these cumulative traversal times are greater than the total navigation time.

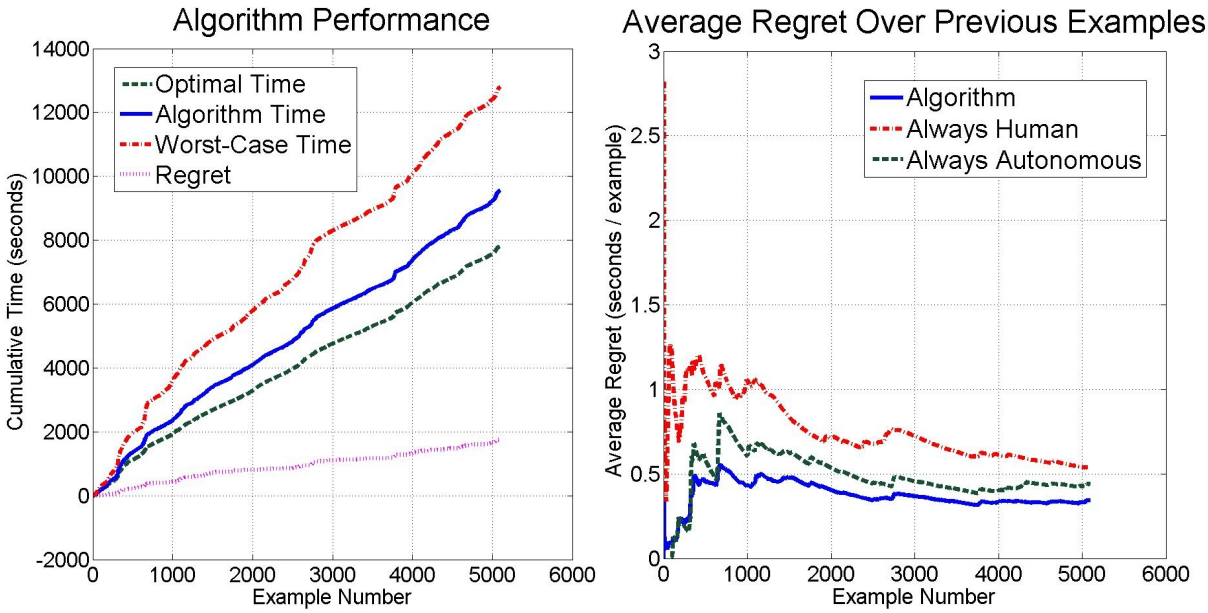


Figure 6.2: Online operator selection performance: cumulative navigation time for our algorithm and various alternatives (left) and the average regret of our algorithm over previous examples compared to alternatives (right).

6.3.2 Online Overhead Data Selection

We also show how the same technique can be used to deal with scenarios where limited high-resolution overhead data is available to aid the robot in navigating through an environment. The Digital Terrain Elevation Data (DTED) level of an overhead elevation data set specifies its density of coverage. DTED level 3 overhead data is available for a majority of the world but is so sparse that in most cases it adds very little to the features that can be generated from overhead imagery. Meanwhile, higher resolution overhead data can be used to produce more accurate traversal cost estimates that the UGV can use for better prior path computation but often require expensive and time-consuming aerial surveying and a large amount of bandwidth if remotely supplied to the vehicle. In scenarios where there is either limited time to gather that data or limited bandwidth for wireless transmission of the data to the vehicle during navigation, the vehicle must decide online how to best utilize the availability of data for upcoming navigation. In such situations, our algorithm can be extended to allow the robot learn to identify the situations where it will most benefit from high resolution data in order to allocate it to areas that maximize its impact.

We simulated this scenario by analyzing sets of multi-waypoint logged runs from a field test at Fort Carson in Colorado on sets of courses using DTED levels 3, 4 and 5 overhead data. The candidates for each waypoint in this case were the choice of density of aerial data for an area bounding that path segment. The candidate selection system therefore had the goal of learning a mapping from the average of feature values (computed as described earlier) within the segment's bounding box to the average traversal speed for the vehicle over that segment of the path using each candidate type of data.

While DTED 5 data almost always resulted in the best performance, we simulated a scenario where high-density data is available for only a fraction of all segments: a maximum of 20%

availability for DTED 5 and 30% availability for DTED 4.

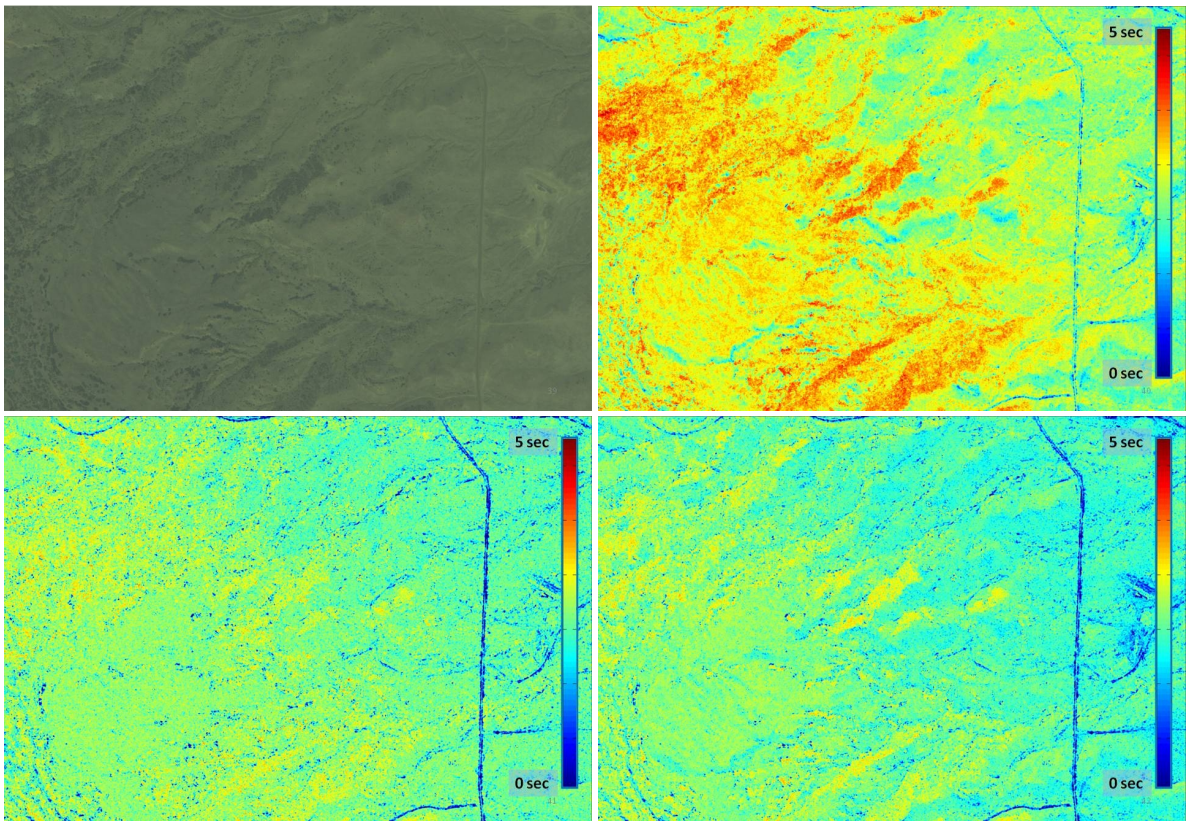


Figure 6.3: Aerial image of sample terrain for data selection experiments is shown in top-left. Estimated traversal time in seconds per meter is shown for DTED 3, 4 and 5 data at top-right, bottom-left and bottom-right respectively. As expected, DTED 5 data shows large improvements in navigation speed for difficult terrain but does not provide much benefit on roads and open fields.

At each step we used a linear program to optimize the allocations of remaining data availability using the predicted performance on all remaining segments from the learned models for each candidate at that time. Selections at each step were based on the initial step of this locally computed optimal allocation. To avoid having to do integer programming, we chose the candidate with the highest allocation at the first step.

The course and estimated rate of progress using each data source predicted by the trained model appear in Figure 6.3. Quantitative results for this scenario appear in Figure 6.4 and Table 6.2.

Our algorithm shows a clear improvement over naive or random approaches for both scenarios with quickly-converging regret properties.

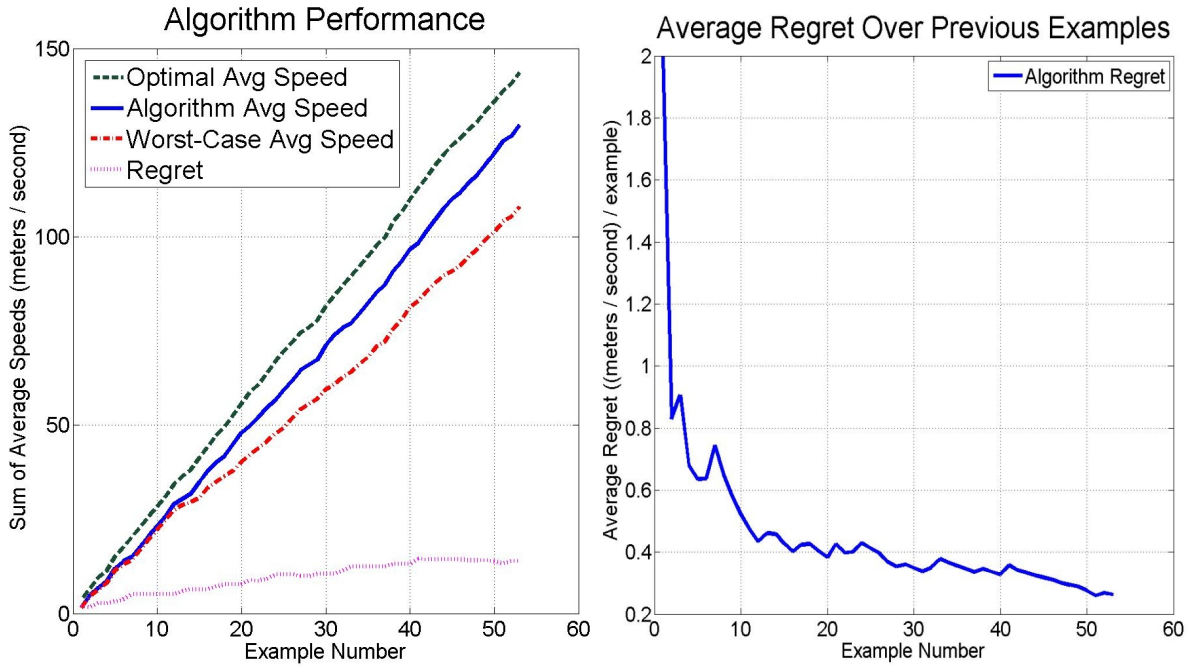


Figure 6.4: Overhead data selection performance: sum of average navigation speed over each path segment for our algorithm and various alternatives (left) and the average regret of our algorithm over previous segments (right).

Table 6.2: Online Overhead Data Selection Performance

Algorithm	Average Speed (meters / second)	Percent Improvement over Random
Online Algorithm	2.45	5.60
Optimal	2.71	16.81
Worst-Case	2.04	-12.07
Random Data Source	2.32	0.00

Chapter 7

Conclusions

This thesis presents a series of online learning techniques that enable a robot to better handle navigation in difficult and unstructured natural environments. The online learning approaches for perception significantly improve the range of a robot's near-range perception system, allowing it to navigate faster and more safely. The online novelty and change detection systems allow a mobile robot to identify potentially dangerous situations in order to request the aid of a human *before* it gets into trouble. The online candidate selection system allows a robot to trade-off between multiple modes of operation, learning to improve its utilization of the different modes with experience.

We demonstrate through both offline and online testing how combining the adaptive performance of these algorithms with the inherent mobility of a capable UGV can lead to more efficient navigation of complex environments. Finally, we present theoretical justification for our approaches as well as a variety of extensions to our algorithms that will further their impact on the field of mobile robotics.

The combination of such techniques can overcome the limitations of offline methods by improving the effectiveness and range of the robot's perception system, dramatically reducing the number of mission-ending errors by identifying potentially hazardous unfamiliar situations, reliably detecting unexpected changes in previously traversed environments and allowing better utilization of the availability of limited human assistance. Such capabilities will increase the reliability and robustness of mobile robot systems and will be a step towards enabling more UGVs to be fielded in real-world applications.

This section presents a short summary of each of the main research topics, reviews the core contribution of each component, and offers some future directions for this work.

7.1 Summary and Contributions

7.1.1 Improved Perception in Unfamiliar Domains

Chapter 4 presented a self-supervised online learning algorithm to learn and infer between different types of data sources that vary in density, reliability, and scope. By applying the scoped

learning model, we were able to generalize from one type of data source to be able to work with another which may be difficult to generalize to new environments. As a result, we were able to extend the scope of such features to many possible domains without requiring any time-consuming human-supervised training.

We showed how the algorithm can be used to improve the navigation capabilities of unmanned ground vehicles by learning in real-time to interpret overhead and far-range sensor data to predict terrain traversal costs generated from an on-board perception system. We demonstrated this approach through field tests on-board a large robot in complex natural environments. Both online and offline results were given to demonstrate several applications of the algorithm. While performance could be hampered because of limitations in the available features or availability of representative training examples, the use of this algorithm was shown to significantly improve robot navigation performance when compared to using just the baseline hand-tuned system.

While our approaches are related to recent sensor fusion ideas, we were the first to apply this type of self-supervised framework to mobile robot perception in this way. As a result we were able to achieve unparalleled perception range for a mobile robot, demonstrating groundbreaking autonomous navigation performance for unstructured natural terrain over many hundreds of kilometers. Since then, many variations of this approach have been demonstrated in numerous other areas of robotics.

7.1.2 Anytime Online Novelty and Change Detection

A variety of algorithms were presented in Chapter 5 for enabling online novelty and change detection and applications for these algorithms in mobile robotics were explored. The presented novelty detection algorithm addresses two significant limitations of most novelty detection approaches. By operating within a lower-dimensional subspace created by using MDA (rather than techniques such as PCA), this algorithm operates in a feature space that is more conducive to viewing novelty as a distance metric and is therefore more resistant to many of the issues associated with high-dimensional feature spaces. Additionally, this algorithm's adaptive abilities, computational bounds and anytime properties make it a logical choice for many online novelty detection tasks including those valuable for mobile robots.

The presented online novelty detection algorithm was also extended to deal with the problem of online change detection. By using an online scene segmentation system, the change detection system was able to achieve further improvements to accuracy and robustness. As robotic systems continue to improve, incorporating such approaches into these systems can allow earlier and more effective deployment by acting as a safeguard against the inevitable dangers of unfamiliar domains.

Safeguarding a mobile robot is most often handled by the perception system that is engineered to estimate degrees of traversability. As robots encounter unfamiliar environments, these engineered systems lose accuracy as the raw data inputs themselves are no longer in the experience set of the robot. While others have explored approaches for novelty (anomaly) and change detection in other domains, much of their focus was on the offline batch application that would be unsuitable for helping mobile robotics to deal with such situations. We are the first to make

a push into online real-time applications of novelty and change detection for robotics, enabling a system to operate for extended periods of time while updating its model and maintaining accurate real-time performance. This type of long-term learning will be critical to enabling robust real-world uses of mobile robots.

7.1.3 Online Candidate Selection

Chapter 6 presented an online algorithm for dealing with scenarios where the robot must learn to trade-off between multiple operating modes. The proposed approach relies on a bandit-based framework and uses confidence bounds to deal with exploration-exploitation trade-offs. The algorithm was demonstrated on two scenarios relevant to the mobile robotics domain: trading off between autonomous and tele-operator control and strategically utilizing limited high-resolution overhead data to maximize its impact on navigation performance. Such techniques could increase the potential real-world applications of mobile robots by allowing them to adapt in real-time to changing environments and better allocate available resources, including the limited availability of human attention.

Other researchers have studied the idea of intermixing human and teleoperator control (referred to as *sliding autonomy* or adjustable autonomy), but a majority of their approaches apply fixed rules and logic to identify when to transition control. We were the first to approach this problem with a general purpose online framework for modeling the performance of arbitrary modes of operation online using contextual information without any prior assumptions. This more general approach would allow these systems to better extend to new environments without requiring additional human engineering.

7.2 Future Work

There are several areas of future research that can follow from this thesis work.

7.2.1 Additional Applications of Self-Supervised Learning

An obvious future area is exploring how other sub-systems within a mobile robot's autonomy system can benefit from online self-supervised learning techniques where the robot itself can act as an expert to train other parts of the system. While most of the algorithms in this thesis focused on the perception system, many opportunities are available in improving onboard planners online. For example, a mobile robot's local planner, often in charge of maneuvering the vehicle to be able to keep it in line with a global path, is notoriously difficult to tune because it must take into account the vehicle's mobility and interaction with the environment. When the environment changes due to rain or different type of terrain for example, the models that are required for accurate planning could also change significantly. Being able to refine these models online using real-time performance could allow robotic systems to adapt robustly to changing conditions.

7.2.2 Novelty and Change Detection

The novelty and change detection systems described in Chapter 5 have shown promising results but more work remains to be done to get these systems to a point where they can be fully fielded on a real-world mobile robot system.

Various improvements are likely to further help performance. The features used for both the novelty and change detection systems were generated by the perception system primarily for the purposes of traversal cost estimation. There are likely additional features that are not currently utilized that may be more applicable for measuring novelty. Feature limitations are an especially large concern for the change detection work as it entirely utilized the Gator platform whose sensing capabilities were limited compared to the UPI robots (the Gators were equipped with only one SICK laser scanner and one camera, providing a limited sensing density and field of view). Furthermore, the novelty detection system would surely benefit from an online scene segmentation component just as the change detection system did.

Also, since segmentation has been shown to be a key to precise change detection, improving the quality and consistency of the segmentation system output can significantly improve performance. Scene segmentation in computer vision has received an incredible amount of attention which we may be able to leverage within our system. For example, influencing 3D segmentation with the results of a robust 2D segmentation computed on the camera could significantly help consistency.

Finally, the system's sensitivity to registration error has proven to be a limiting factor in some difficult scenarios. While normal GPS systems will always have some uncertainty in their estimates, online vision-based registration techniques have the potential to reduce registration error to unprecedented levels, likely resulting in a sharp performance boost.

The ROC curves in Figures 5.10 and 5.19 also raise several questions. First, these results measure performance over a large number of voxels. While such a volume-based evaluation is the fairest way to evaluate performance on such a broad number of tests, the size of an object is obviously not always directly related to its potential threat to the robot (and therefore importance in evaluation).

Furthermore, a concern that arises from these figures is the false positive rate required to achieve accurate identification of novel or changed locations. As in many machine learning problems, this trade-off between true and false positives is a key concern. While the correct balance is highly dependent on the particular domain the robot is operating within, various steps can be taken to improve performance to necessary levels while limiting false positives.

For example, if a change is detected and a human operator identifies it as a false positive, an additional novelty detection instance can be used within the same run to identify and ignore similar false positives from further consideration. Any irrelevant changes would be added to the novelty detection model and all future situations would only be considered for potential changes if they are novel with respect to the remembered false positives. The highest impact approach to improving this trade-off, however, would likely be the intelligent uncertainty resolution approach proposed in in the next section.

The systems described in this thesis have been analyzed thoroughly both qualitatively and

quantitatively, but the true measure of their performance is their impact on a robot during operation. To enable this, a robust tele-operation system for allowing a remote human operator to control the robot needs to be developed. Such a system would enable the operator to either send desired commands back to the robot or to take control of the robot directly. It would then be possible to more thoroughly evaluate the performance of the system by also measuring the amount of human attention required and how much such a system improves overall system safety and performance.

7.2.3 Intelligent Uncertainty Resolution

Another high-impact line of research is a form of *intelligent uncertainty resolution.* When utilizing novelty detection and candidate selection systems such as the ones proposed here in an autonomous navigation scenarios, the availability and cost of a human operator is often high. The system must therefore consider beyond just estimated performance or uncertainty in choosing the candidate for each situation.

A logical extension would be to incorporate an intelligent uncertainty resolution technique to minimize the potential number of human queries during navigation. With a functioning novelty or change detection system, it is important to consider the benefit of resolving uncertainty, presumably through a human query. For example, if a novel object blocks a primary route to the goal and the next best alternative has a much higher cost, then the potential benefit of involving a human exceeds the cost (see an example scenario in Figure 7.1). Similarly, a candidate selection system choosing between autonomous and human control can avoid unnecessary human involvement in situations that can be relatively easily circumnavigated autonomously, allowing larger human utilization at more critical situations. By considering the possible impact on the optimal global path of both possible extremes (first assuming the location is trivially traversable and then assuming it is completely untraversable), the system can choose to simply avoid novel or potentially difficult situations that cannot significantly improve our metrics.

Such an approach provides several key advantages. As mentioned previously, it enables us to better utilize human attention during hybrid control systems such as the candidate selection framework. Possibly of even higher impact, such an approach would allow a higher sensitivity within novelty or change detection systems without unacceptably high human time commitments. Because only a minority of situations will have the potential to significantly influence our relevant metrics (and therefore trigger a human query), the system could tolerate more false positives without the potential of heavily burdening a human supervisor.

Finally, each of these systems was evaluated individually. It would be beneficial to see how an autonomy system that fully utilizes the state of the art in learning approaches, both a priori (such as training through demonstration as shown in [112]) and online, can perform in diverse, unstructured environments.

Figure 7.1: An uncertainty resolution technique that can distinguish between situations such as those above could further improve the impact of techniques such as those in this thesis. In the left image, resolving the uncertainty at this pinch point through a human query can potentially save significant time and travel distance. In the right image, the outcome of a human query cannot significantly effect the quality of the path.

Appendix A

Bayesian Linear Regression

Bayesian Linear Regression is a simple yet powerful algorithm whose properties make it highly suitable for online learning applications. The online learning systems for improving perception (Chapter 4) and the online candidate selection systems (Chapter 6) in this thesis use Bayesian Linear Regression as their core learner.

If a linear model is sufficient for an online learning task, Bayesian Linear Regression is a great algorithm for several reasons. First, it allows the model to be updated efficiently in real-time without recomputation or consideration of old data. This means that the performance scales well in online learning tasks meant to operate for long periods of time rather than growing in complexity with the amount of training data considered. In fact, the computation time for prediction depends only on the number of features and is completely independent of the number of examples incorporated into the model. This powerful property allows the system to represent a potentially infinite amount of training data without having to remember these examples for future use. This is a significant advantage over approaches like Gaussian Processes where the the cubic dependency on the number of examples considered makes using large training sets infeasible for online tasks.

Additionally, if the problem has the property that several varying quality estimates may be available for each example, Bayesian Linear Regression enables *un-learning* of old data so that the example can be re-learned with the more accurate estimate (in this case the old features and labels need to be remembered so that the model can properly adjust when un-learning them).

Finally, Bayesian Linear Regression can also provide variance estimates when making predictions which are often useful for applications such as mobile robotics. The combination of these properties and ease of implementation make it a versatile tool for numerous online learning scenarios.

We will assume for the rest of this chapter that x represents features for an incoming example and y represents the prediction for those features. Assuming that d is the dimensionality of these feature vectors, then $x \in \Re^d$ and $y \in \Re$. Training examples arrive in the form of (x, y) pairs while examples for prediction will only have the features x and the system tries to make an estimate of the output, $\tilde{y}$, given all previously seen data.

We assume that x and y are related through some weights θ as follows:

$$y = \theta^T x + \epsilon \tag{A.1}$$

where ϵ is an assumed Gaussian noise of the form:

$$\epsilon \sim N(0, \sigma^2) \tag{A.2}$$

The goal of Bayesian Linear Regression is to maintain online an estimate distribution for θ that takes into account all seen data and takes into account any assumptions of noise.

In this chapter we will derive the algorithm and discuss in detail its various properties and typical uses. Section A.1 will present relevant background on Gaussian distributions that will prove useful throughout the rest of this chapter. Section A.2 will discuss the initialization process for Bayesian Linear Regression, and the process for training and prediction for incoming examples will be discussed in Sections A.3 and A.4 respectively. Finally, Section A.5 will explain a typical implementation and use of Bayesian Linear Regression through simple pseudocode.

A.1 Basic Gaussian Properties

The density function of a Gaussian distribution can typically be parameterized in two ways. The more familiar parameterization, called *Moment Parameterization*, represents a distribution for θ as follows:

$$p(\theta) \propto e^{-\frac{1}{2}(\theta-\mu)^T \Sigma^{-1} (\theta-\mu)} \tag{A.3}$$

where μ is the *mean* of the distribution and Σ is the *covariance matrix*, representing the interdependencies between the variables of θ.

An alternative parameterization, called *Natural Parameterization*, represents a distribution for θ as follows:

$$p(\theta) \propto e^{-\frac{1}{2}\theta^T P \theta + J^T \theta} \tag{A.4}$$

where the two parameterizations are related as follows:

$$P = \Sigma^{-1} \tag{A.5}$$
$$J = \Sigma^{-1}\mu = P\mu \tag{A.6}$$
$$\mu = P^{-1} J \tag{A.7}$$
$$\tag{A.8}$$

where P is often referred to as the *precision matrix* and J is a vector sometimes called the *information vector*.

For clarity, we will sometimes refer to the mean estimate for θ as $\hat{\theta}$ rather than μ.

A.2 Initialization

We assume that $p(\theta)$ is initialized to some initial distribution $p_0(\theta) = N(\mu, \Sigma)$ that captures the initial belief about θ.

If one has no prior knowledge about our model, this may simply be a zero-mean Gaussian with an initial covariance matrix *diag(σ^2)*. If, on the other hand, there is an initial belief about the model, one may initialize μ to that belief and set the covariance matrix Σ to represent the appropriate level of confidence in that initial model (more detail on this approach can be found in Section A.5). As described in Chapter 4, such a technique can be used to initialize a model to a state learned from many previous examples and then continue to adjust it an more information about the current environment becomes available.

A.3 Training

We begin with our prior distribution $p_0(\theta)$. We then receive an example (x, y). Through Bayes Rule, we know that:

$$p(\theta|y, x) \propto p(y|x, \theta)p(\theta) \tag{A.9}$$

This can be seen as revising the posterior distribution for $p(\theta)$ in light of a new Gaussian likelihood term.

To perform these update steps in an efficient fashion, it is easier to operate on the Natural Parameterization of the distribution $p(\theta)$. We assume we have the initial distribution:

$$p(\theta) \propto e^{-\frac{1}{2}\theta^T P\theta + J^T\theta} \tag{A.10}$$

and the new likelihood term:

$$p(y|x, \theta) \propto e^{-(\frac{y-\theta^T x}{2\sigma^2})^2} \tag{A.11}$$

Expanding the exponent in equation A.11 we get:

$$p(y|x, \theta) \propto e^{\frac{-y^2}{2\sigma^2} + \frac{2y\theta^T x}{2\sigma^2} - \frac{(\theta^T x)^2}{2\sigma^2}} \tag{A.12}$$

The third term in the exponent can be further expanded as follows:

$$-\frac{(\theta^T x)^2}{2\sigma^2} = -\frac{(\theta^T x)(x^T\theta)}{2\sigma^2} = -\frac{\theta^T(xx^T)\theta}{2\sigma^2} \tag{A.13}$$

giving us the likelihood term:

$$p(y|x,\theta) \propto e^{\frac{-y^2}{2\sigma^2}+\frac{y\theta^T x}{\sigma^2}-\frac{\theta^T(xx^T)\theta}{2\sigma^2}} \tag{A.14}$$

From equation A.9, we combine and rearrange equations A.10 and A.14 as follows:

$$p(\theta|y,x) \propto p(\theta)p(y|x,\theta) \tag{A.15}$$

$$p(\theta|y,x) \propto e^{-\frac{1}{2}\theta^T P\theta + J^T\theta} e^{\frac{-y^2}{2\sigma^2}+\frac{2y\theta^T x}{2\sigma^2}-\frac{\theta^T(xx^T)\theta}{2\sigma^2}} \tag{A.16}$$

$$p(\theta|y,x) \propto e^{\frac{-y^2}{2\sigma^2}} e^{-\frac{1}{2}(\theta^T P\theta+\frac{\theta^T(xx^T)\theta}{\sigma^2})+(J^T\theta+\frac{y\theta^T x}{\sigma^2})} \tag{A.17}$$

$$p(\theta|y,x) \propto e^{\frac{-y^2}{2\sigma^2}} e^{-\frac{1}{2}\theta^T(P+\frac{xx^T}{\sigma^2})\theta+(J+\frac{yx}{\sigma^2})^T\theta} \tag{A.18}$$

The first term in A.18 gets absorbed by the normalizer term, giving us:

$$p(\theta|y,x) \propto e^{-\frac{1}{2}\theta^T(P+\frac{xx^T}{\sigma^2})\theta+(J+\frac{yx}{\sigma^2})^T\theta} \tag{A.19}$$

Therefore, to update the model θ at some timestamp t to take into account a new training example (x_t, y_t), we simply need to update P and J as follows:

$$P_{t+1} = P_t + \frac{x_t x_t^T}{\sigma^2} \tag{A.20}$$

$$J_{t+1} = J_t + \frac{y_t x_t}{\sigma^2} \tag{A.21}$$

Computational complexity for incorporating a new training example into the model is simply $O(d^2)$ regardless of how many examples were already incorporated into the model.

Intuitively, what's happening here is that as we see more examples, the precision matrix P grows, and Σ therefore shrinks, meaning that we are reducing our uncertainty about new examples (see Equation A.27 for how to make variance predictions).

One exciting capability that this form allows is *unlearning* an example by simply subtracting out its likelihood term:

$$P = P - \frac{xx^T}{\sigma^2} \tag{A.22}$$

$$J = J - \frac{yx}{\sigma^2} \tag{A.23}$$

As shown in Chapter 4, this is highly beneficial in fields such as robotics when you may receive many estimates for the same quantity that vary in accuracy. As more accurate estimates

of y become available, the system can easily undo the effects of the old, noisy example and relearn the example with the more accurate measurement.

A.4 Prediction

To perform prediction, we convert back to Moment Parameterization of θ:

$$\hat{\theta} = P^{-1}J \tag{A.24}$$
$$\Sigma = P^{-1} \tag{A.25}$$

We can now use $\hat{\theta}$ to predict y for a given x:

$$\begin{aligned} p(y|x) &= \theta^T x + \epsilon \\ E[y] &= E[\theta^T x + \epsilon] \\ E[y] &= E[\theta^T x] + E[\epsilon] \\ E[y] &= E[\theta^T x] + 0 \\ E[y] &= \hat{\theta}^T x \end{aligned} \tag{A.26}$$

We can also make an variance estimate (a measure of confidence in our prediction) as follows:

$$\begin{aligned} Var[y] &= Var[\theta^T x + \epsilon] \\ Var[y] &= Var[\theta^T x] + Var[\epsilon] \\ Var[y] &= Var[\theta^T x] + \sigma^2 \\ Var[y] &= E[(\theta^T x)^2] - E[\theta^T x]^2 + \sigma^2 \\ Var[y] &= E[x^T(\theta\theta^T)x] - x^T\hat{\theta}\hat{\theta}^T x + \sigma^2 \\ Var[y] &= x^T(E[\theta\theta^T] - \hat{\theta}\hat{\theta}^T)x + \sigma^2 \\ Cov[\theta] &= E[\theta\theta^T] - E[\theta]E[\theta]^T \\ Cov[\theta] &= E[\theta\theta^T] - \hat{\theta}\hat{\theta}^T \\ Var[y] &= x^T(Cov(\theta))x + \sigma^2 \\ Var[y] &= x^T\Sigma_\theta x + \sigma^2 \end{aligned} \tag{A.27}$$

Note that Σ is dependent only on x and is independent of y. The variance estimate therefore measures how well the example's features x are represented by previously seen examples' features that were incorporated into the model.

It is now visible that the computational complexity for prediction on a new example is only $O(d)$ for mean prediction and $O(d^2)$ for variance prediction.

A.5 Sample Use for Online Learning Task

When using Bayesian Linear Regression, one would constantly switch between the Natural Parameterization Form (P and J) for training and the Moment Parameterization form ($\hat{\theta}$ and Σ) for prediction. As shown earlier, computational time per new example within each of these modes is short.

The most computationally expensive step transitioning between training and prediction which requires $O(d^3)$ time due to the inversion of a matrix. Fortunately, in applications such as mobile robotics, examples for training and for prediction often come in large batches, so the penalties for changing forms are relatively small.

Typical use of Bayesian Linear Regression for an online learning task can be seen in Algorithm 6. We assume for this example that the model is initialized to some prior uncertainty defined by σ^2 and that training and prediction examples arrive in batches. Each time the system needs to switch from one mode to another, it switches which representation it uses for the distribution of θ in order to facilitate the mode it is in.

Algorithm 6 Sample use of Bayesian Linear Regression for Online Learning Task

1: **given:** Assumed prior variance, σ^2.
2: **initialize:** $\mu \leftarrow \vec{\mathbf{0}}$; $\Sigma \leftarrow diag(\sigma^2)$
3: **loop**
4: **Convert to Natural Parameterization for training batch**
5: $P \leftarrow \Sigma^{-1}$
6: $J \leftarrow P\mu$
7: **for** each training example (x, y) in training batch **do**
8: **Update model to incorporate example**
9: $P \leftarrow P + \frac{xx^T}{\sigma^2}$
10: $J \leftarrow J + \frac{yx}{\sigma^2}$
11: **end for**
12: **Convert to Moment Parameterization for prediction batch**
13: $\Sigma \leftarrow P^{-1}$
14: $\hat{\theta} \leftarrow P^{-1}J$
15: **for** each prediction example (x) in prediction batch **do**
16: **Compute estimate** $\tilde{y}$ **for features** x
17: $\tilde{y} \leftarrow \hat{\theta}^T x$
18: $Var[\tilde{y}] \leftarrow x^T \Sigma x + \sigma^2$
19: **end for**
20: **end loop**

In this example the initially untrained system is initialized to a zero-mean model with a large initial prior variance (line 2). If one desires the system to continue training from a previously trained model, one would simply initialize μ and Σ to that model's last computed μ and Σ.

In some scenarios, it may be useful to initialize a model to a previously trained state but to put a low confidence in that initial model so that it will be quickly replaced as new information

is acquired. One such scenario is when using a far-range perception system such as the one described in Chapter 4. Such a system could be trained in a set of known environments and must now operate in an unknown environment. In such scenarios the previously computed values of μ are obviously more accurate than initializing to a zero-mean model, but we have a low confidence in this model and want it to be quickly replaced by newly acquired knowledge.

Because the confidence in a model is entirely captured by Σ, one can replicate the behavior of a less trained system by simply initializing Σ to that of such a system. For example, in such a scenario one could initialize μ to the best-known prior model and initialize Σ to a covariance matrix stored after several minutes of training on a previous traversal. The model will then be gradually replaced at a rate appropriate for a model trained for the amount of time captured by Σ.

In general, lower values within Σ signify more confidence in the model meaning more evidence will be required to modify it significantly. It may therefore be advantageous to make sure values within Σ do not fall below some constant value so that the model is able to quickly adapt to new terrain throughout operation.

Appendix B

Self-Organizing Lists

A core component of the anytime online novelty detection system described in Chapter 5 is the list maintenance strategy used to continually re-order the stored examples to maintain a low average query time throughout navigation. This section looks in more detail into the area of self-organizing lists, explains the specific formulation of the problem we deal with, presents some related problems and proofs, and evaluates various approaches for the specific online case we are dealing with.

B.1 Dictionary Problem

A well-known problem referred to as the *dictionary* or *list search problem* involves maintaining and adjusting a set of items in response to an intermixed sequence of queries taking the following form:

access(i)	:	Locate item i in the set.
insert(i)	:	Insert item i in the set.
delete(i)	:	Delete item i from the set.

A common way to solve this problem is to represent the set as an unsorted list. Accessing an item involves scanning through the list sequentially from the front until the item is located[1]. Likewise, inserting an item involves scanning through the list to verify that it does not already exist and then inserting it in the rear. Deleting an item involves scanning from the front of the list to find the item and then removing it from the list.

In addition to performing these three operations, we may rearrange the list throughout these processes by moving encountered elements to earlier portions of the list[2]. Such a strategy can

[1]We assume that if i is not in the list, **access**(i) will insert it into the list.

[2]We assume that the list can be represented as a linked list so that rearranging elements in the list can be performed in constant time.

speed up future operations.

Based on the definitions of the operations above, the costs of the various operations are as follows. Accessing or deleting the i^{th} item in the list costs i (based on the time to find that element). Inserting a new item costs $i+1$ where i is the size of the list before insertion. Immediately after an access or insertion of an item i, i can be moved at no cost to any position closer to the front of the list.

One of the most intuitive approaches for list organization is the *move-to-front* (MF) approach where after accessing or inserting an item, it is moved to the front of the list, without changing the relative order of the other items.

In the following discussion, n shall be used to denote the maximum number of items ever in the set at one time and m will denote the total number of operations. Also, for any algorithm A let c_A be the cost of a single operation to A and C_A be the cumulative cost of all accesses.

Among algorithms that do no rearranging of the list, it can be easily shown that ordering the elements in decreasing frequency (DF) of accesses minimizes the cumulative access cost. A well-known result by Bentley and McGeoch [9] showed that for a fixed list of n items on which only accesses are permitted, for any sequence of accesses s, $C_{MF} \leq 2 * C_{DF}(s)$. This means that the total cost when using the move-to-front algorithm with sequence s is no worse than twice the cost of using the optimally ordered static list for this sequence.

We now present a result by Sleator and Tarjan that generalizes the result by Bentley and McGeoch to non-static lists [113]. While the original proof deals with a more general case of list search optimality, we focus on the formulation that is relevant to our problem.

For any Algorithm A and any sequence of operations s, let $C_A(s)$ be the total cost of all the operations and let $F_A(s)$ be the number of free exchanges (immediately following an access or insertion).

Theorem 1. *For the list-search problem, for any Algorithm A and any sequence of operations s starting with the empty set, $C_{MF}(s) \leq 2C_A(s) - m - F_A(s)$.*

Proof. This proof makes use of the concept of a *potential function*. If algorithm A and MF were run in parallel on s, a potential function maps a configuration of A's and MF's lists into a real number ϕ. If we perform an operation that takes time t and changes the configuration to one with potential ϕ', we define the *amortized time* of the operation to be $t + \phi' - \phi$. The amortized time of an operation is therefore its running time plus the increase it causes in the potential.

If we perform a sequence of operations such that the ith operation takes actual time t_i and has amortized time a_i, then we have the following relationship:

$$\begin{aligned} a_i &= t_i + \phi_i - \phi_{i-1} \\ t_i &= a_i + \phi_{i-1} - \phi_i \\ \sum_i t_i &= \sum_i a_i + \phi_0 - \phi_m \end{aligned}$$

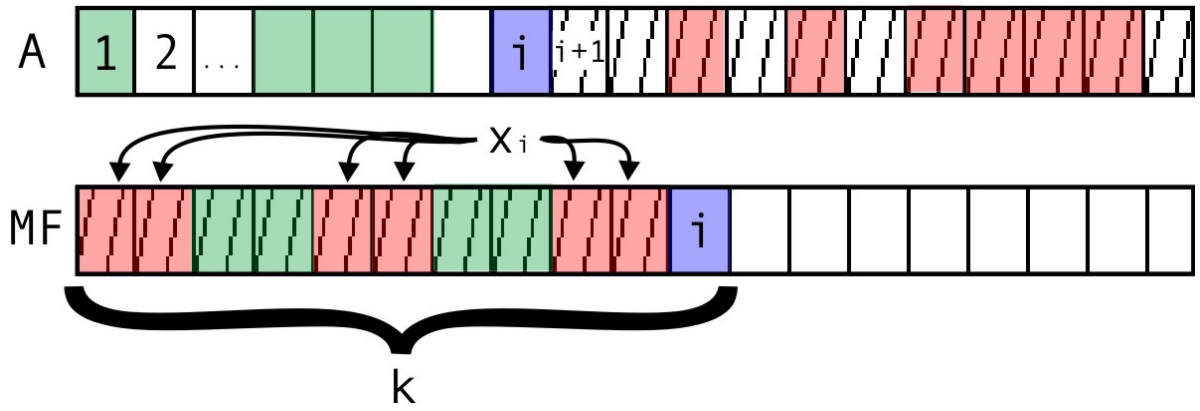


Figure B.1: Example states of lists under algorithms A and MF (for the list search problem) prior to an access call for element i. For the initial k elements of list A, those shaded in red compose set x_i, those elements that represent inversions relative to element i. Elements shaded in green appear before i in both lists. The striped region contains cells that could possibly contain inversions relative to i.

where ϕ_0 is the initial potential and ϕ_m is the final potential. We can therefore estimate the total running time by choosing a potential function and bounding ϕ, ϕ_m, and a_i for each i.

For this theorem, we use as the potential function the number of inversions in MF's list with respect to A's list. For any two lists containing the same items, an $inversion$ in one list with respect to the other is an unordered pair of items, i and j, such that i occurs anywhere before j in one list and anywhere after j in the other.

This proof shall show that the amortized time for MF to access item i is at most $2i - 1$ and the amortized time for MF to insert an item into a list of size i is at most $2(i + 1) - 1$, where we identify an item by its position in A's list. Furthermore, the amortized time charged to MF when A does an exchange is -1. Because an item is identified by its position in A's list, an access or insertion would then have amortized time $2c_A - 1$, where c_A is the cost of the operation to A. The -1's, one per operation, sum to $-m$. Proving these properties will show that the amortized time of MF is bounded by the true cost of A.

The initial configuration, ϕ_0, has zero potential since the initial lists are empty, and the final configuration, ϕ_m has a nonnegative potential, so the actual cost to MF of a sequence of operations, $\sum_i t_i$, is bounded by the sums of the operations' amortized times, $\sum_i a_i$. The sum of the amortized times is in turn bounded by the right-hand side of the inequality we wish to prove.

All that remains is for us to bound the amortized times of the operations. Consider an access by A to an item i (see Figure B.1). Let k be the position of i in MF's list and let x_i be the number of items that precede i in MF's list but follow i in A's list (shown with red shade). The number of items preceding i in both lists is therefore $k - 1 - x_i$ (shown with a blue shade).

Moving i to the front of MF's list will create $k - 1 - x_i$ new inversions and destroy x_i other inversions (see Figure B.2). Since the lookup cost to MF is k, the amortized time for the operation (cost plus the increase in the number of inversions) is $k+(k-1-x_i)-x_i = 2(k-x_i)-1$. $k - x_i$ is the number of elements that are before i in both lists (the elements shaded in green in Figure B.1). Because only $i - 1$ items precede i in A's list, $k - x_i \leq i$. The amortized time for the access is therefore at most $2i - 1$. Since the cost of this access to algorithm A is i, the

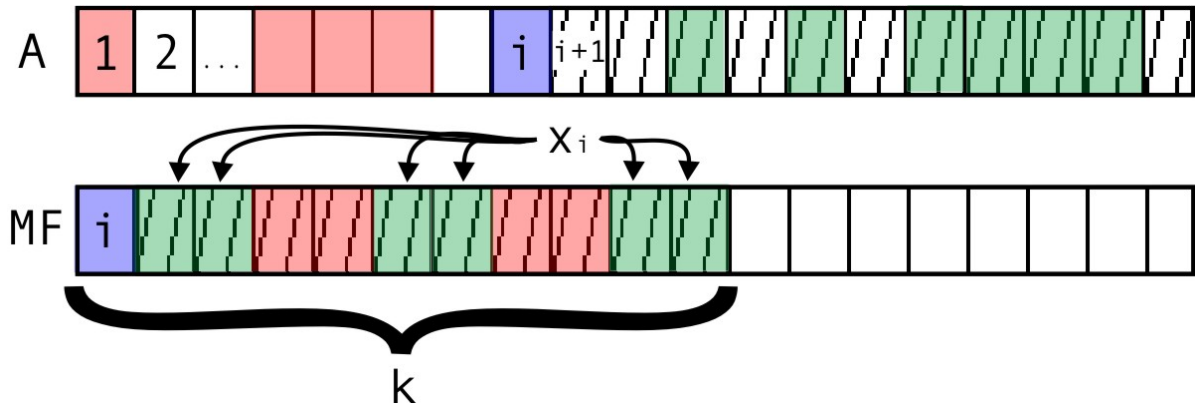


Figure B.2: State of the lists shown in Figure B.1 after a move-to-front action on element i in MF. The inversion states of the first $k-1$ elements in Figure B.1 are inverted, meaning $k-1-x_i$ new inversions are created and x_i of the original inversions are deleted. The striped region is the same as that shown in Figure B.1.

amortized time of this access to MF is at most $2c_A - 1$.

The argument for the amortized time for insertions and deletions apply almost identically.

An exchange action (moving a queried element to earlier in the list) by algorithm A has zero cost to MF, so the amortized time of an exchange is simply the increase in the number of inversions caused by the exchange. This increase is at most -1 because each movement of element i to earlier in list A will decrease the number of inversions (since i is now at the front of MF).

The total cost of $MF(s)$ for the list search problem is therefore bounded by $2C_A(s) - m - F_A(s)$, meaning it's within a factor of two of the best possible algorithm. □

B.2 Min-Sum Weighted Set Cover

We are interested in exploring a variant of the dictionary problem that is relevant to online scenarios such as the novelty detection approach described in Chapter 5. We will call the offline version of this problem the *min-sum weighted set cover* problem and its online counterpart the *online min-sum weighted set cover* problem. An explanation of how our problem is related to the set cover problem and its variations (and hence the chosen name) will be discussed in Section B.3.

Rather than searching for a single quantity at each time step, in this online scenario each time step t provides an example $\mathbf{x}_t$ consisting of some vector of features and the system must make a prediction $f_t(\mathbf{x}_t)$ where $f : \mathcal{X} \rightarrow \{novel, not\text{-}novel\}$.

The result of this function depends on the influence on this new example, $\mathbf{x}_t$, from all previously stored examples $\{\mathbf{x}\}_{1\ldots t-1}$ computed using the positive symmetric kernel function $k(x_i, x_j)$ and some novelty threshold γ as follows:

$$f_t(\mathbf{x}_t) = \begin{cases} \textit{not-novel} & \text{if } \sum_{i=1}^{t-1} k(\mathbf{x}_i, \mathbf{x}_t) \geq \gamma \\ \textit{novel} & \text{otherwise} \end{cases} \tag{B.1}$$

This novelty query problem therefore is one of accumulating the non-negative contributions from all previously stored examples onto this query to see if the sum exceeds the threshold γ. If the previously stored examples are maintained in a list, the ordering of those examples can have a large impact on computational complexity. While novel incoming examples (where $f(x) < \gamma$) will require processing each element in the stored examples list regardless of its ordering, for non-novel examples (where $f(x) \geq \gamma$) the function can return the moment the cumulative contribution reaches γ (since future contributions cannot change the outcome of the query). If a majority of queries are non-novel, as is the case for many novelty detection tasks, an intelligent list maintenance can lead to fewer required evaluations and therefore have a large impact on overall system performance.

For each query, there will be some index i in the stored examples list at which the query is fully resolved. This parallels to directly searching for the element at index i in the dictionary problem. Our objective is to minimize the sum of these first indices over all examples by reaching the novelty threshold γ for each query (*covering* that query) after evaluating as few stored examples as possible. This would minimize the average computation time per query for a decision to be made.

B.3 Related Problems

Closer examination reveals that this formulation of the cumulative-contribution version of the dictionary problem resembles several well-studied problems in computer science.

B.3.1 Set Cover Problem

The set cover problem is a classical question in computer science. You are given several sets that may have some elements in common. You must select a minimum number of these sets so that the sets you have picked contain all the elements that in the union of all the sets. This problem was shown to be NP-hard and has received heavy focus in the field of approximation algorithms. Johnson [55] showed in 1974 that the greedy approach gives a $\ln n$ approximation to the optimal set covering and Feige later showed that this is a tight bound [31].

Our list maintenance problem can be viewed as a variation of the set cover problem. The incoming query, or set of queries, parallel the set of elements that need to be covered in the set cover problem. Each stored example in turn act as the sets themselves, each contributing some amount to the incoming queried elements. *Covering* an element in our case means satisfying the novelty threshold γ using some number of stored examples, or sets.

There are two added points of complexity to the min-sum weighted set cover problem over the basic set cover problem. First, each stored example may not fully cover an incoming query, but rather contribute some non-negative amount toward its novelty threshold γ, so each query will need to be *covered* by multiple sets. Second, the order of the chosen sets is also important because we want to minimize the average number of stored examples that we must evaluate before resolving a query.

We can, however, show that the min-sum weighted set cover problem is NP-hard by reducing the standard set cover problem to our min-sum weighted set cover problem.

Theorem 2. *Min-Sum Weighted Set Cover is NP-hard.*

Proof. Given a set cover problem, it can be transferred to a min-sum weighted set cover problem as follows. Each element to be covered in the set cover problem becomes one of the queries to be covered and each set becomes a stored example that fully covers (provides the entire required contribution) the elements within that set and provides zero contribution to the remaining queries. The solution to this min-sum weighted set cover problem would then also be a solution to the original set cover problem. This polynomial time reduction from the set cover problem proves that the min-sum weighted version of set cover is also NP-hard. □

B.3.2 Min-Sum Set Cover Problem

Feige introduced a variation of the set cover problem called the *min-sum set cover* problem (a close relative of the min-sum vertex cover problem) [30]. Like in the set cover problem, the input to min-sum set cover is a collection of n sets that jointly cover m elements. The output is a linear ordering on the sets where at every time step from 1 to n exactly one set is chosen. For every element, this induces a first time step at which it is first covered. The objective is to find a linear arrangement of the sets that minimizes the sum of these first time steps over all elements.

Feige showed in [30] that the greedy approach (iteratively adding the element which maximally increases the objective value) approximates min-sum set cover within a ratio of 4 and the results from [8] show that finding a closer approximation is NP-hard.

Even though it does not consider partial coverage of queries, the min-sum set cover problem is closely related to the cumulative contribution set cover problem because it incorporates the time at which each element is covered into the metric. We call our problem the min-sum weighted set cover problem because it closely resembles this min-sum set cover problem but adds a weight onto the contributions from each set onto each element.

B.3.3 Pipelined Set Cover Problem

A generalization of the set cover problem called the *pipelined set cover problem* was discussed in [88]. This formulation is similar to that of the min-sum set cover problem except that the cost of using each set does not need to be uniform. Here the costs of using each set are weighted rather than the impacts of each set on each element like as in our case. This problem is especially relevant to database applications where the cost of evaluating a set is proportional to the number of elements in that set, rather than constant as discussed in the previous set cover problems.

The greedy approximation was shown to also approximate the optimal solution for the pipelined set cover problem within a factor of 4.

B.4 Submodularity

Many problems can be characterized by the property of *submodularity* [36, 64, 90]. These problems satisfy the intuitive diminishing returns property: adding elements to the solution helps more early on and less over time. More formally, consider a set function F which maps subsets $\mathcal{A} \subseteq \mathcal{V}$ of a finite set $\mathcal{V}$ to the real numbers. F is called *submodular* if, for all $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{V}$ and $\mathcal{S} \in \mathcal{V} \setminus \mathcal{B}$ it holds that

$$F(\mathcal{A} \cup \mathcal{S}) - F(\mathcal{A}) \geq F(\mathcal{B} \cup \mathcal{S}) - F(\mathcal{B}) \tag{B.2}$$

In other words, adding set $\mathcal{S}$ to a small set $\mathcal{A}$ helps at least as much as when it's added to a superset $\mathcal{B}$.

If one can prove that a problem is submodular, then using a greedy algorithm for that problem can provide a strong performance guarantee:

Theorem 3 (From [90])**.** *If F is a non-decreasing submodular function, then the solution $\mathcal{A}_{greedy}$ returned by the greedy algorithm satisfies*

$$F(\mathcal{A}_{greedy}) \geq (1 - \frac{1}{e})OPT \tag{B.3}$$

where OPT is the value obtained by an optimal solution.

Therefore, the greedy algorithm is guaranteed to obtain a solution which achieves at least a constant fraction of $(1 - \frac{1}{e}) \approx 63\%$ of the optimal value. Because many NP-hard problems are submodular, this powerful result shows that a simple greedy approach can often closely approximate an optimal solution that is extremely difficult to find.

For our cumulative contribution set cover problem, we want to fully cover a set of incoming queries $\mathcal{X} \subseteq \mathcal{V}$ (in our case this means resolving those queries) using a set of stored examples $\mathcal{A} \subseteq \mathcal{V}$ and a symmetric kernel function k. The function F we are maximizing is therefore:

$$F(\mathcal{A}) = \sum_{x \in \mathcal{X}} min \left\{ \gamma, \sum_{a \in \mathcal{A}} k(a, x) \right\} \tag{B.4}$$

In other words, for the set of queries $\mathcal{X}$, we want to maximize the total contribution from the stored examples $\mathcal{A}$ where the cumulative contribution for each query is capped at the novelty threshold γ.

As shown in Theorem 2, finding the optimal subset and ordering for list $\mathcal{A}$ that optimizes this function is NP-hard, but if we can show that F is submodular, then using a greedy approach to populate $\mathcal{A}$ will provide an efficient way to find a solution to the cumulative contribution set cover problem that closely approximates the optimal solution.

Theorem 4. *The value function for the cumulative contribution set cover problem (function F in equation B.4) is submodular.*

Proof. We begin by considering F_x which optimizes over a single query element $x \in \mathcal{X}$:

$$F_x(\mathcal{A}) = min\left\{\gamma, \sum_{a\in\mathcal{A}} k(a,x)\right\} \tag{B.5}$$

Assume we are given sets $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{V}$, $\mathcal{S} \in \mathcal{V} \setminus \mathcal{B}$ and a set of queries $\mathcal{X} \subseteq \mathcal{V}$ that we want to cover. From equations B.2 and B.5 we need to prove the following inequality:

$$F_x(\mathcal{A}\cup\mathcal{S}) - F_x(\mathcal{A}) \geq F_x(\mathcal{B}\cup\mathcal{S}) - F_x(\mathcal{B}) \tag{B.6}$$

Let $\mathcal{R} = \mathcal{B} \setminus \mathcal{A}$, the set of elements that are in $\mathcal{B}$ but not in $\mathcal{A}$. Since $\mathcal{A}\cup\mathcal{R} = \mathcal{B}$, we can now rewrite equation B.6 as follows:

$$F_x(\mathcal{A}\cup\mathcal{S}) - F_x(\mathcal{A}) \geq F_x(\mathcal{A}\cup\mathcal{R}\cup\mathcal{S}) - F_x(\mathcal{A}\cup\mathcal{R}) \tag{B.7}$$

Let $\beta = \sum_{a\in\mathcal{A}} k(a,x)$, the baseline total contribution (before cropping it at γ) from the stored elements of $\mathcal{A}$ before adding the contribution from the elements in $\mathcal{S}$. Using equation B.5 we can now expand and evaluate the left side of equation B.7 as follows:

$$\begin{aligned} F_x(\mathcal{A}\cup\mathcal{S}) - F_x(\mathcal{A}) &= min\left\{\gamma, \sum_{a\in\mathcal{A}} k(a,x) + \sum_{s\in\mathcal{S}} k(s,x)\right\} \\ &\quad -min\left\{\gamma, \sum_{a\in\mathcal{A}} k(a,x)\right\} \end{aligned} \tag{B.8}$$

$$F_x(\mathcal{A}\cup\mathcal{S}) - F_x(\mathcal{A}) = min\left\{\gamma, \beta + \sum_{s\in\mathcal{S}} k(s,x)\right\} - min\left\{\gamma, \beta\right\} \tag{B.9}$$

$$F_x(\mathcal{A}\cup\mathcal{S}) - F_x(\mathcal{A}) = \begin{cases} 0 \text{ if} & \beta \geq \gamma \\ \gamma - \beta \text{ if} & \beta > \gamma - \sum_{s\in\mathcal{S}} k(s,x) \text{ and } \beta < \gamma \\ \sum_{s\in\mathcal{S}} k(s,x) \text{ if} & \beta \leq \gamma - \sum_{s\in\mathcal{S}} k(s,x) \end{cases} \tag{B.10}$$

Expanding and evaluating the right side of equation B.7 through similar fashion gives us the following:

$$\begin{aligned} F_x(\mathcal{A}\cup\mathcal{R}\cup\mathcal{S}) - F_x(\mathcal{A}\cup\mathcal{R}) &= min\left\{\gamma, \sum_{a\in\mathcal{A}} k(a,x) + \sum_{r\in\mathcal{R}} k(r,x) + \sum_{s\in\mathcal{S}} k(s,x)\right\} \\ &\quad -min\left\{\gamma, \sum_{a\in\mathcal{A}} k(a,x) + \sum_{r\in\mathcal{R}} k(r,x)\right\} \end{aligned} \tag{B.11}$$

$$F_x(\mathcal{A}\cup\mathcal{R}\cup\mathcal{S}) - F_x(\mathcal{A}\cup\mathcal{R}) = min\left\{\gamma, \beta + \sum_{r\in\mathcal{R}} k(r,x) + \sum_{s\in\mathcal{S}} k(s,x)\right\} \tag{B.12}$$

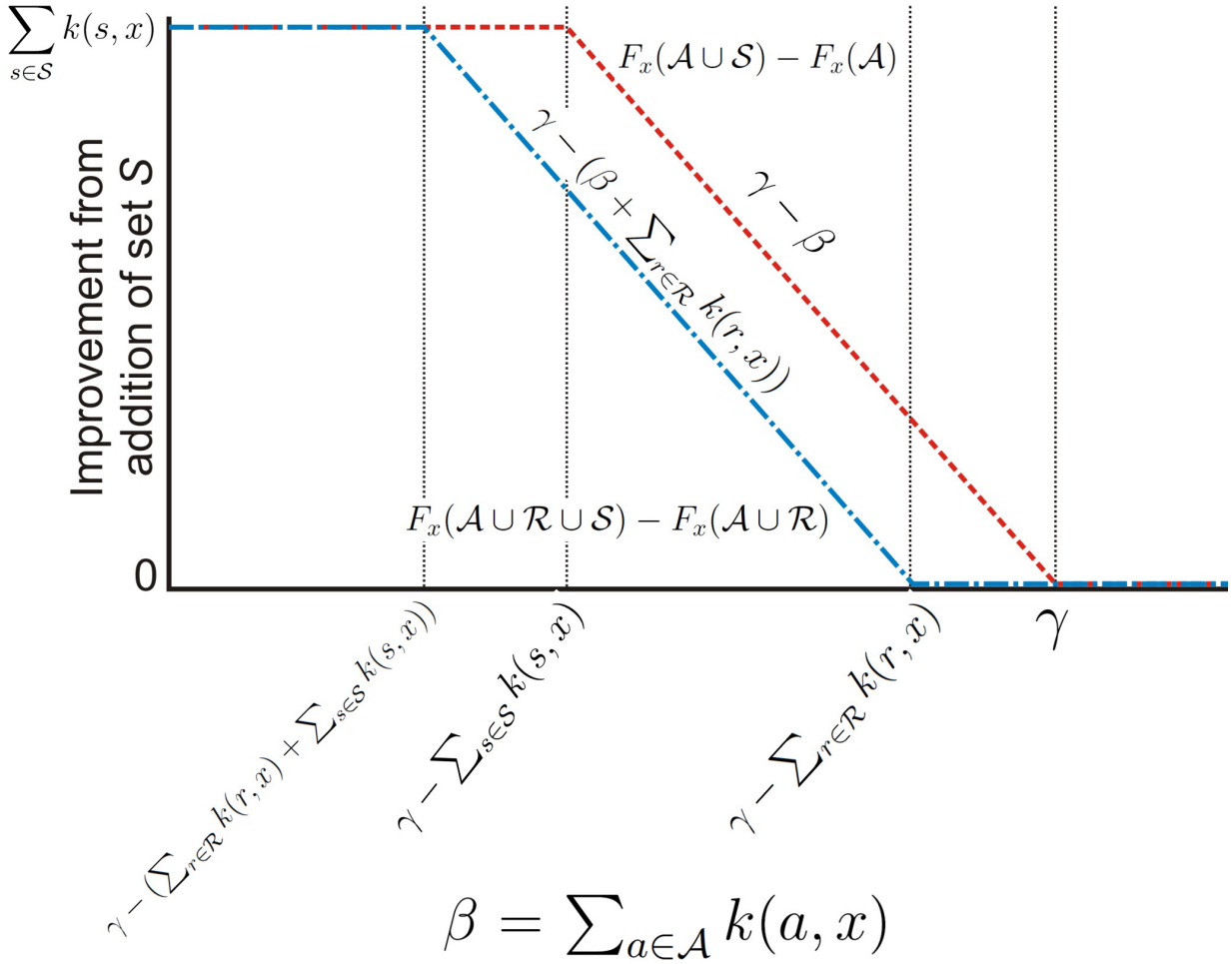


Figure B.3: Visualization of each side of equation B.7. $F_x(\mathcal{A}\cup\mathcal{S}) - F_x(\mathcal{A})$ and $F_x(\mathcal{A}\cup\mathcal{R}\cup\mathcal{S}) - F_x(\mathcal{A}\cup\mathcal{R})$ as a function of the initial contribution β are shown in red and blue respectively. The gained value for the left side of equation B.7 dominates the gained value of the right side for all values of β, proving that B.5 is submodular.

$$-min\left\{\gamma, \beta + \sum_{r\in\mathcal{R}} k(r,x)\right\}$$

$$= \begin{cases} 0 \text{ if} & \beta \geq \gamma - \sum_{r\in\mathcal{R}} k(r,x) \\ \gamma - (\beta + \sum_{r\in\mathcal{R}} k(r,x)) \text{ if} & \beta > \gamma - (\sum_{r\in\mathcal{R}} k(r,x) + \sum_{s\in\mathcal{S}} k(s,x)) \text{ and} \\ & \beta < \gamma - \sum_{r\in\mathcal{R}} k(r,x) \\ \sum_{s\in\mathcal{S}} k(s,x) \text{ if} & \beta \leq \gamma - (\sum_{r\in\mathcal{R}} k(r,x) + \sum_{s\in\mathcal{S}} k(s,x)) \end{cases} \tag{B.13}$$

Using the values found in equations B.10 and B.13, the improvement gained for each side of equation B.7 can be seen in Figure B.3.

Because the gained value for the left side of equation B.7 (shown in red) dominates the gained value of the right side (shown in blue) for all values of β, adding a set $\mathcal{S}$ to a smaller existing set always has at least as much impact as when $\mathcal{S}$ is added to a larger set. F_x from equation B.5 is therefore submodular.

From equations B.4 and B.5, it is easy to see that:

$$F(\mathcal{A}) = \sum_{x \in \mathcal{X}} F_x(\mathcal{A}) = \sum_{x \in \mathcal{X}} min \left\{ \gamma, \sum_{a \in \mathcal{A}} k(a, x) \right\} \tag{B.14}$$

Because F_x is submodular and submodularity is closed under nonnegative linear combinations, it follows that F is submodular as well.

□

Following Theorems 3 and 4, a greedy algorithm for the offline cumulative contribution set cover problem would provide a solution within a constant factor of the optimal.

However, taking advantage of this property for the online scenario we are interested in is impractical. The direct goal here is to resolve queries as quickly as possible, not necessarily to identify the set of stored examples that would do so in the most efficient way. In order to execute the greedy algorithm for this problem, for each query we must evaluate the impact of each stored example at each step in the ordering. With such a high overhead for this step, it is computationally simpler to just compute the influence of every stored example on every incoming query, at which point the order of evaluation no longer matters.

Furthermore, if we wanted to replicate the behavior of the greedy algorithm on the entire set of examples seen so far, we would have to measure and store the impact of each element in our stored list on each element in the query list. This is impractical to maintain, especially if the stored elements change over time as they would under our implementation.

B.5 Online Submodular Minimization

For a specific class of submodular online resource allocation problems, Streeter and Golovin [125] introduced an online algorithm whose worst-case performance approaches that of the offline greedy approximation algorithm asymptotically. Specifically, the online algorithm's 4-regret (regret with respect to the factor of 4 approximation from the offline algorithm) for the min-sum set cover problem described in Section B.3.2 approaches zero as the number of incoming queries approaches infinity.

For their online setting, we are fed a sequence $\{f_1, f_2, \ldots, f_n\}$ of jobs one at a time. Prior to receiving job f_i, we must specify a schedule S_i that defines the sequence of sets that will be used to satisfy this job, where the number of sets is at most T.

One of the key components of this algorithm makes use of the *experts problem*. In this problem one has k experts, each of whom gives out a piece of advice at each step. At each step i, one must select an expert e_i whose advice to follow. Following the advice of expert j on step i yields a reward r_j^i. At the end of step i, the value of the reward x_j^i for each expert j is made public, and can be used as the basis for making choices on subsequent steps. One's regret at the end of n steps is then measured with respect to the best single expert over all examples seen:

$$\max_{1 \leq j \leq k} \left\{ \sum_{i=1}^{n} x_j^i \right\} - \sum_{i=1}^{n} x_{e_i}^i \tag{B.15}$$

Because some randomized decision-making algorithms exist whose regret grows sub-linearly in the number of steps, by picking experts using such an algorithm, one can be guaranteed to obtain asymptotically an average reward that is as large as the maximum reward that could have been obtained following the advice of any fixed expert for all n days. The online algorithm from [125] makes use of this surprising property as shown in Algorithm 7. In this case, the randomized weighted majority algorithm [74] can be used for each expert to achieve this zero-regret property.

Algorithm 7 Online algorithm from [125] for submodular resource allocation problems

1: **given:** Integer T, expert algorithms $\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_T$
2: **for** $i = 1$ to n: **do**
3: For each t, $1 \leq t \leq T$, use $\mathcal{E}_t$ to select an action a_t^i
4: Select the schedule $S_i = \{a_1^i, a_2^i, \ldots, a_T^i\}$
5: Receive the job f_i
6: For each t, $1 \leq t \leq T$, and each action $a \in \mathcal{A}$, feed back the gained function value as the payoff $\mathcal{E}_t$ would have received by choosing action a.
7: **end for**

The algorithm runs T distinct copies of expert algorithms: $\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_T$. Just before job f_i arrives, each expert algorithm $\mathcal{E}_t$ selects an action a_t^i. The set order used on job f_i is then $S_i = \{a_1^i, a_2^i, \ldots, a_T^i\}$. The payoff that $\mathcal{E}_t$ associates with action a is the increase in the function value achieved after using action a.

It is shown in [125] that Algorithm 7 with randomized weighted majority as the subroutine expert algorithm has an expected regret of $O(\sqrt{Tn \ln |\mathcal{A}|})$ in the worst case, a sub-linear regret that asymptotically approaches the offline regret as $n \to \infty$.

While this is an impressive theoretical bound, using such an algorithm for the online scenario described in Chapter 5 would again be impractical. The prohibitive step can be found on line 6 of Algorithm 7. In this step we must update each expert with the gained function value for using each potential stored example at that step. Because varying the position of the evaluation can change the resulting contribution, if we have m stored examples and therefore set $T = m$, we would incur an m^2 cost for every incoming query. With such a high overhead for this bookkeeping step, it would again be computationally simpler to just compute the influence of every stored example on every incoming query, at which point the order of evaluation no longer matters.

Bibliography

[1] Pieter Abbeel, Adam Coates, Michael Montemerlo, and Andrew Y. Ng andSebastian Thrun. Discriminative training of kalman filters. In *Proceedings of Robotics: Science and Systems*, 2005. 4.1

[2] A. Angelova, L. Matthies, D. Helmick, G. Sibley, and P. Perona. Learning to predict slip for ground robots. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, 2006. 4.1

[3] D. Anguelov, B. Taskar, V. Chatalbashev, D. Koller, D. Gupta, G. Heitz, and A. Ng. Discriminative learning of markov random fields for segmentation of 3D scan data. In *CVPR*, pages II: 169–176, 2005. 5.4

[4] Brenna Argall, Yang Gu, Brett Browning, and Maria Manuela Veloso. The first segway soccer experience: Towards peer-to-peer human-robot teams. In *In Proceedings First Annual Conference on Human-Robot Interactions*, March 2006. 6.1

[5] P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *The Journal of Machine Learning Research*, 3:397–422, 2003. 6.2.2

[6] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning, 47*, 2(3):235–256, 2002. 6.2.1

[7] James Andrew Bagnell, David Bradley, David Silver, Boris Sofman, and Anthony Stentz. Learning for autonomous navigation: Advances in machine learning for rough terrain mobility. *IEEE Robotics and Automation Magazine*, 17(2):74–84, June 2010. 3.3

[8] Bar-Noy, Halldorsson, and Kortsarz. A matched approximation bound for the sum of a greedy coloring. *IPL: Information Processing Letters*, 71, 1999. B.3.2

[9] J. L. Bentley and C. A. Cole. Worst-case analyses of self-organizing sequential search heuristics. *20th Annual Allerton Conf. on Communication, Control, and Computing*, pages 452–461, 1982. B.1

[10] Richard Bishop. Intelligent vehicle applications worldwide. *IEEE Intelligent Systems*, 15 (1):78–81, 2000. 6.1

[11] David Blei, James Bagnell, and Andrew McCallum. Learning with scope, with application to information extraction and classification. In *Proceedings of the 2002 Conference on Uncertainty in Artificial Intelligence*, June 2002. 4.2.1, 1

[12] B. Bodta and R. Camden. Technology readiness level 6 and autonomous mobility. In *Proceedings of the SPIE, Volume 5422, Unmanned Ground Vehicle Technology VI*, 2004.

1, 1.1

[13] David M. Bradley, Ranjith Unnikrishnan, and James Bagnell. Vegetation detection for driving in complex environments. In *ICRA*, pages 503–508. IEEE, 2007. 3.3.1, 3.6

[14] David M. Brann, David A. Thurman, and Christine M. Mitchell. Human interaction with lights-out automation: a field study. In *In Proceedings of the 1996 Symposium on Human Interaction and Complex Systems*, pages 276–283, 1996. 6.1

[15] Colin Campbell and Kristin P. Bennett. A linear programming approach to novelty detection. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, *NIPS*, pages 395–401. MIT Press, 2000. 5.1

[16] Guo Cao, Xin Yang, and Zhihong Mao. A two-stage level set evolution scheme for man-made objects detection. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, 2005. 4.1

[17] J. Carsten, A. Rankin, D. Ferguson, and A. Stentz. Global path planning on board the mars exploration rovers. *Aerospace Conference, 2007 IEEE*, pages 1–11, March 2007. ISSN 1095-323X. 6.1

[18] D.W. Casbeer, R.W. Beard, T.W. McLain, Sai-Ming Li, and R.K. Mehra. Forest fire monitoring with multiple small UAVs. *American Control Conference, 2005. Proceedings of the 2005*, pages 3530–3535 vol. 5, June 2005. ISSN 0743-1619. 6.1

[19] Jennifer Casper and Robin R. Murphy. Human-robot interactions during the robot-assisted urban search and rescue response at the world trade center. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 33(3):367–385, 2003. 6.1

[20] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`. 2

[21] Amin P. Charaniya, Roberto Manduchi, and Suresh K. Lodha. Supervised parametric classification of aerial lidar data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2004. 4.1

[22] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Mach. Intell*, 24(5):603–619, 2002. 5.4

[23] R. Craig Coulter, Anthony Stentz, Paul G. Keller, Gary K. Shaffer, William Red L. Whittaker, Barry Brummit, and William Burky. A system for telerobotic control of servicing tasks in a nuclear steam generator. Technical Report CMU-RI-TR-90-24, Robotics Institute, Pittsburgh, PA, December 1990. 6.1

[24] R. Craig Coulter, Anthony Stentz, P. Keller, and William Red L. Whittaker. A telerobotic solution for tool insertion tasks in nuclear servicing. In *Remote Systems Session of the ANS Winter Meeting*, November 1991. 6.1

[25] Hendrik Dahlkamp, Adrian Kaehler, David Stavens, Sebastian Thrun, and Gary R. Bradski. Self-supervised monocular road detection in desert terrain. In Gaurav S. Sukhatme, Stefan Schaal, Wolfram Burgard, and Dieter Fox, editors, *Robotics: Science and Systems*. The MIT Press, 2006. 4.1

[26] Varsha Dani, Thomas P. Hayes, and Sham M. Kakade. Stochastic linear optimization under bandit feedback. In *21st Annual Conference on Learning Theory - COLT 2008, Helsinki, Finland, July 9-12, 2008*, pages 355–366. Omnipress, 2008. 6.2.3

[27] M Bernardine Dias, Balajee Kannan, Brett Browning, Edward Jones, Brenna Argall, Malcolm Frederick Dias, Marc B. Zinck, Maria Manuela Veloso, and Anthony Stentz. Sliding autonomy for peer-to-peer human-robot teams. In *10th International Conference on Intelligent Autonomous Systems 2008*, July 2008. 6.1

[28] Gregory A. Dorais, R. Peter Bonasso, David Kortenkamp, Barney Pell, and Debra Schreckenghost. Adjustable autonomy for human-centered autonomous systems on mars, April 12 1998. 6.1

[29] R. O. Duda and P. E. Hart. *Pattern Classification*. John Wiley and Sons, 2000. 4.3.1, 5.1, 5.2.2

[30] Feige, Lovasz, and Tetali. Approximating min sum set cover. *ALGRTHMICA: Algorithmica*, 40, 2004. B.3.2

[31] Uriel Feige. A threshold of ln n for approximating set cover. *J. ACM*, 45(4):634–652, 1998. B.3.1

[32] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, September 2004. 5.4

[33] David Ferguson and Anthony Stentz. Using interpolation to improve path planning: The field D* algorithm. In *Journal of Field Robotics*, volume 23, pages 79–101. John Wiley & Sons, February 2006. 1.1, 3.3.1, 6.1

[34] Terrence W. Fong. *Collaborative Control: A Robot-Centric Model for Vehicle Teleoperation*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, November 2001. 6.1

[35] Terrence W. Fong, Chuck Thorpe, and Charles Baur. Multi-robot remote driving with collaborative control. *IEEE Transactions on Industrial Electronics*, 2003. 6.1

[36] Satoru Fujishige. *Submodular Functions and Optimization*. Elsevier, 2 edition, 2005. B.4

[37] Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin. *Bayesian Data Analysis*. Chapman and Hall/CRC, 2004. 4.2.1, 4.2.2, 4.3.3

[38] B.K. Ghosh and P.K. Sen. *Handbook of sequential analysis*. Marcel Dekker, 1991. 6.2.1

[39] A.R. Girard, A.S. Howell, and J.K. Hedrick. Border patrol and surveillance missions using multiple unmanned air vehicles. *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, 1:620–625 Vol.1, Dec. 2004. ISSN 0191-2216. 6.1

[40] S. Goldberg, Maimone, and L. Matthies. Stereo vision and rover navigation software for planetary exploration. In *Proceedings of the IEEE Aerospace Conference*, 2002. 1, 1.1

[41] Aleksy Golovinskiy and Thomas Funkhouser. Min-cut based segmentation of point clouds. Princeton University. 5.4

[42] Michael A. Goodrich and Alan C. Schultz. Human-robot interaction: A survey. *Foundations and Trends in Human-Computer Interaction*, 1(3):203–275, 2007. 6.1

[43] Richard Grace, V.E. Byrne, D.M. Bierman, J.-M. Legrand, D. Gricourt, B.K. Davis, J.J. Staszewski, and B. Carnahan. A drowsy driver detection system for heavy vehicles. In *Proceedings of the 17th Digital Avionics Systems Conference*, volume 2, pages I36/1 – I36/8, 2001. 6.1

[44] James P. Gunderson and Worthy N. Martin. Effects of uncertainty on variable autonomy in maintenance robots. In *In Workshop on Autonomy Control Software*, pages 26–34, 1999. 6.1

[45] Paul Hayton, Bernhard Schölkopf, Lionel Tarassenko, et al. Support vector novelty detection applied to jet engine vibration spectra. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, *NIPS*, pages 946–952. MIT Press, 2000. 5.1

[46] Frederik W. Heger and Sanjiv Singh. Sliding autonomy for complex coordinated multi-robot tasks: Analysis & experiments. In Gaurav S. Sukhatme, Stefan Schaal, Wolfram Burgard, and Dieter Fox, editors, *Robotics: Science and Systems*. The MIT Press, 2006. ISBN 0-262-69348-8. 6.1

[47] Henry Hexmoor. A cognitive model of situated autonomy. *Lecture Notes in Computer Science*, 2112:325–, 2001. ISSN 0302-9743. 6.1

[48] H. Hoffmann. Kernel PCA for novelty detection. *Pattern Recognition*, 40(3):863–874, March 2007. 5.1

[49] Eric Horvitz, Andy Jacobs, and David Hovel. Attention-sensitive alerting. In Kathryn B. Laskey and Henri Prade, editors, *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pages 305–313, S.F., Cal., July 30–August 1 1999. Morgan Kaufmann Publishers. 6.1

[50] Daniel Huber, Anuj Kapuria, Raghavendra Rao Donamukkala, and Martial Hebert. Parts-based 3d object classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 04)*, June 2004. 5.4

[51] S. Huwer and H. Niemann. Adaptive change detection for real-time surveillance applications. In *Third IEEE International Workshop on Visual Surveillance*, pages 37–45, Dublin, July 2000. IEEE. 5.1

[52] L. D. Jackel, Eric Krotkov, Michael Perschbacher, Jim Pippine, and Chad Sullivan. The DARPA LAGR program: Goals, challenges, methodology, and phase I results. *Journal of Field Robotics*, 23(11-12):945–973, 2006. 1, 1.1

[53] Nathalie Japkowicz, Catherine Myers, and Mark A. Gluck. A novelty detection approach to classification. In *IJCAI*, pages 518–523, 1995. 5.1

[54] A. E. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 21(5):433–449, May 1999. 5.4

[55] D. S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences, Academic*, 9, 1974. B.3.1

[56] Edward Jones, Brett Browning, M Bernardine Dias, Brenna Argall, Maria Manuela Veloso, and Anthony Stentz. Dynamically formed heterogeneous robot teams perform-

ing tightly-coordinated tasks. In *International Conference on Robotics and Automation*, pages 570 – 575, May 2006. 6.1

[57] M. I. Jordan and Y. Weiss. *Graphical models: Probabilistic inference*. MIT Press, 2002. 4.2.1

[58] Sham M. Kakade and Andrew Y. Ng. Online bounds for bayesian algorithms. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 641–648. MIT Press, Cambridge, MA, 2005. 4.2.1

[59] Alonzo Kelly, Anthony Stentz, Omead Amidi, et al. Toward reliable off road autonomous vehicles operating in challenging environments. *The International Journal of Robotics Research*, 25(5-6):449–483, 2006. 1, 1.1, 3.3.1

[60] D. Kim, J. Sun, S. Oh, J. Rehg, and A.Bobick. Traversability classification using unsupervised on-line visual learning for outdoor robot navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2006. 4.1

[61] Jyrki Kivinen, Alex J. Smola, and Robert C. Williamson. Online learning with kernels. In Thomas G. Dietterich, Suzanna Becker, and Zoubin Ghahramani, editors, *NIPS*, pages 785–792. MIT Press, 2001. 5, 5.2.3, 5.2.3, 5.10

[62] Thomas Knudsen and Allan Aasbjerg Nielson. Detection of buildings through multivariate analysis of spectral, textural, and shape based features. In *Proceedings of IGARSS*, 2004. 4.1

[63] Kolmogorov and Zabih. What energy functions can be minimized via graph cuts. *IEEET-PAMI: IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26, 2004. 5.4.4

[64] Andreas Krause. *Optimizing Sensing: Theory and Applications*. PhD thesis, Carnegie Mellon University, 2008. B.4

[65] Eric Krotkov, Reid Simmons, Fabio Cozman, and Sven Koenig. Safeguarded teleoperation for lunar rovers: From human factors to field trials. In *In Proc. IEEE Planetary Rover Technology and Systems Workshop*, 1996. 6.1

[66] Alexandre Krupa, Michel de Mathelin, Christophe Doignon, Jacques Gangloff, Guillaume Morel, Luc Soler, and Jacques Marescaux. Development of semi-autonomous control modes in laparoscopic surgery using automatic visual servoing. *Lecture Notes in Computer Science*, 2208:1306–??, 2001. ISSN 0302-9743. 6.1

[67] Y. Kuno, T. Watanabe, Y. Shimosakoda, and S. Nakagawa. Automated detection of human for visual surveillance system. In *International Conference on Pattern Recognition*, pages III: 865–869, 1996. 5.1

[68] T.L. Lai. Adaptive treatment allocation and the multi-armed bandit problem. *The Annals of Statistics*, pages 1091–1114, 1987. 6.2.1

[69] TL Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985. 6.2.1

[70] Jean-Francois Lalonde, Nicolas Vandapel, Daniel Huber, and Martial Hebert. Natural terrain classification using three-dimensional ladar data for ground robot mobility. *Journal of Field Robotics*, 23(1):839 – 861, November 2006. 3.3.1

[71] J. Langford and T. Zhang. The epoch-greedy algorithm for contextual multi-armed bandits. *Advances in Neural Information Processing Systems*, 2007. 6.2.1

[72] Yann LeCun, Urs Muller, Jan Ben, Eric Cosatto, and Beat Flepp. Off-road obstacle avoidance through end-to-end learning. In *Proceedings of the Neural Information Processing Systems*, 2005. 4.1

[73] David Lieb, Andrew Lookingbill, and Sebastian Thrun. Adaptive road following using self-supervised learning and reverse opticalflow. In *Proceedings of Robotics: Science and Systems*, June 2005. 4.1

[74] N. Littlestone and M.K. Warmuth. The weighted majority algorithm. *Foundations of Computer Science*, 0:256–261, 1989. B.5

[75] A. Lookingbill, J. Rogers, D. Lieb, J. Curry, and S. Thrun. Reverse optical flow for self-supervised adaptive autonomous robot navigation. *International Journal of Computer Vision*, 74(3):287–302, September 2007. 4.1

[76] Rajiv T. Maheswaran, Milind Tambe, Pradeep Varakantham, and Karen L. Myers. Adjustable autonomy challenges in personal assistant agents: A position paper. In Matthias Nickles, Michael Rovatsos, and Gerhard Weiß, editors, *Agents and Computational Autonomy*, volume 2969 of *Lecture Notes in Computer Science*, pages 187–194. Springer, 2003. ISBN 3-540-22477-7. 6.1

[77] A. Makarov, J. M. Vesin, and M. Kunt. Intrusion detection using extraction of moving edges. In *International Conference on Pattern Recognition*, pages A:804–807, 1994. 5.1

[78] Larry M. Manevitz and Malik Yousef. One-class SVMs for document classification. *Journal of Machine Learning Research*, 2:139–154, 2001. 5.1

[79] Markos Markou and Sameer Singh. Novelty detection: a review - part 1: statistical approaches. *Signal Processing*, 83(12):2481–2497, 2003. 5.1

[80] Markos Markou and Sameer Singh. Novelty detection: a review - part 2: neural network based approaches. *Signal Processing*, 83(12):2499–2521, 2003. 5.1

[81] Stephen Marsland, Ulrich Nehmzow, and Jonathan Shapiro. On-line novelty detection for autonomous mobile robots. *Robotics and Autonomous Systems*, 51(2-3):191–206, 2005. 5.1

[82] J.-F. Mas. Monitoring land-cover changes: A comparison of change detection techniques. 1999. 5.1

[83] J.C. McCall and M.M. Trivedi. Driver behavior and situation aware brake assistance for intelligent vehicles. *Proceedings of the IEEE*, 95(2), 2007. 6.1

[84] Jeff Michels, Ashutosh Saxena, and Andrew Y. Ng. High speed obstacle avoidance using monocular vision and reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2006. 4.1

[85] T. Minka. A family of algorithms for approximate bayesian inference, 2001. 4.2.2

[86] G. Mori, S. J. Belongie, and J. Malik. Shape contexts enable efficient retrieval of similar shapes. In *CVPR*, pages I:723–730, 2001. 5.4

[87] D. M. Muchoney and B. N. Haack. Change detection for monitoring forest defoliation. *Photogrammetric Engineering and Remote Sensing*, 60(10):1243–1251, October 1994. 5.1

[88] Munagala, Babu, Motwani, and Widom. The pipelined set cover problem. In *ICDT: 10th International Conference on Database Theory*, 2005. B.3.3

[89] Daniel Munoz, Nicolas Vandapel, and Martial Hebert. Onboard contextual classification of 3-d point clouds with learned high-order markov random fields. The Robotics Institute, Carnegie Mellon University. 5.4

[90] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions. *Mathematical Programming*, 14:265–294, 1978. B.4, 3

[91] Hugo Vieira Neto and Ulrich Nehmzow. Visual novelty detection with automatic scale selection. *Robotics and Autonomous Systems*, 55(9):693–701, 2007. 5.1

[92] Bradford Neuman. Segmentation-based online change detection for mobile robots. Technical Report CMU-RI-TR-10-30, Robotics Institute, August 2010. 5.4.1

[93] Andrew Y. Ng and Michael I. Jordan. Convergence rates of the voting gibbs classifier, with application to bayesianfeature selection. In *Proceedings of the International Conference on Machine Learning*, 2001. 4.2.2

[94] I.R. Nourbakhsh, K. Sycara, M. Koes, M. Yong, M. Lewis, and S. Burion. Human-robot teaming for search and rescue. *Pervasive Computing, IEEE*, 4(1):72–79, Jan.-March 2005. ISSN 1536-1268. 6.1

[95] N. Paragios and G. Tziritas. Detection and location of moving objects using deterministic relaxation algorithms. In *International Conference on Pattern Recognition*, pages I: 201–205, 1996. 5.1

[96] Thomas Pilarski, James Bagnell, and Anthony Stentz. Hazard detection for familiar terrains via change detection. Master's thesis, Carnegie Mellon University, 2007. 5.1

[97] H. Robbins. Some aspects of the sequential design of experiments. *Bull. Amer. Math. Soc*, 58(5):527–535, 1952. 6.2.1

[98] Patrick Rowe. *Adaptive Motion Planning for Autonomous Mass Excavation*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, January 1999. 4.1

[99] Salvador Ruiz-Correa, Linda G. Shapiro, Marina Meila, and Gabriel Berson. Discriminating deformable shape classes. In Sebastian Thrun, Lawrence K. Saul, and Bernhard Schölkopf, editors, *NIPS*. MIT Press, 2003. ISBN 0-262-20152-6. 5.4

[100] Jake Ryan, Meng-Jang Lin, and Risto Miikkulainen. Intrusion detection with neural networks. In Michael I. Jordan, Michael J. Kearns, and Sara A. Solla, editors, *Advances in Neural Information Processing Systems*, volume 10. The MIT Press, 1998. 5.1

[101] Paul Scerri, David V. Pynadath, and Milind Tambe. Towards adjustable autonomy for the real world. *Journal of Artificial Intelligence Research*, 17:2002, 2002. 6.1

[102] B. Schölkopf and A. J. Smola. *Learning with Kernels*. The MIT Press, Cambridge, MA,

2002. 5.2.1

[103] Bernhard Schölkopf, Robert C. Williamson, Alex J. Smola, et al. Support vector method for novelty detection. In Sara A. Solla, Todd K. Leen, and Klaus-Robert Müller, editors, *NIPS*, pages 582–588. The MIT Press, 1999. ISBN 0-262-19450-3. 5.1

[104] Jean Scholtz, Jeff Young, Jill L. Drury, and Holly A. Yanco. Evaluation of human-robot interaction awareness in search and rescue. In *International Conference on Robotics and Automation*, pages 2327–2332. IEEE, 2004. 6.1

[105] Chris Scrapper, Ayako Takeuchi, Tommy Chang, Tsai Hong, and Michael Shneier. Using a priori data for prediction and object recognition in an autonomous mobile vehicle. In *Proceedings of the SPIE Aerosense Conference*, April 2003. 4.1

[106] Brennan Peter Sellner, Frederik Heger, Laura Hiatt, Reid Simmons, and Sanjiv Singh. Coordinated multi-agent teams and sliding autonomy for large-scale assembly. *Proceedings of the IEEE - Special Issue on Multi-Robot Systems*, 94(1):1425 – 1444, July 2006. 6.1

[107] Linda G. Shapiro and George C. Stockman. *Computer Vision*. Prentice-Hall, 2001. 5.4

[108] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Conf. Computer Vision and Pattern Recognition*, June 1997. 5.4

[109] N. Z. Shor, Krzysztof C. Kiwiel, and Andrzej Ruszcayǹski. *Minimization methods for non-differentiable functions*. Springer-Verlag New York, Inc., New York, NY, USA, 1985. ISBN 0-387-12763-1. 5.2.3

[110] David Silver, Boris Sofman, Nicolas Vandapel, J. Andrew Bagnell, and Anthony Stentz. Experimental analysis of overhead data processing to support long range navigation. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 2443 – 2450, October 2006. 3.3.2, 4.1, 4.3.1, 4.4.2

[111] David Silver, James (Drew) Bagnell, and Anthony (Tony) Stentz. High performance outdoor navigation from overhead data using imitation learning. In *Robotics Science and Systems*, July 2008. 3.3.2

[112] David Silver, J. Andrew (Drew) Bagnell, and Anthony (Tony) Stentz. Learning from demonstration for autonomous navigation in complex unstructured terrain. *International Journal of Robotics Research*, June 2010. 3.2.2, 3.3.1, 4.1, 5, 7.2.3

[113] Daniel Sleator and Robert Tarjan. Amortized efficiency of list update and paging rules. *CACM: Communications of the ACM*, 28, 1985. 5.2.4, B.1

[114] Boris Sofman, J. Andrew Bagnell, Anthony Stentz, and Nicolas Vandapel. Terrain classification from aerial data to support ground vehicle navigation. Technical report, Robotics Institute, Carnegie Mellon University, August 2005. 3.3.2, 4.1

[115] Boris Sofman, Ellie Lin Ratliff, J. Andrew Bagnell, John Cole, Nicolas Vandapel, and Anthony Stentz. Improving robot navigation through self-supervised online learning. *Journal of Field Robotics*, 23(1), December 2006. 4

[116] Boris Sofman, J. Andrew Bagnell, and Anthony Stentz. Bandit-based online candidate selection for adjustable autonomy. In *7th International Conferences on Field and Service Robotics*, July 2009. 6

[117] Boris Sofman, J. Andrew Bagnell, and Anthony Stentz. Anytime online novelty detection for vehicle safeguarding. In *IEEE International Conference on Robotics and Automation*, May 2010. 5.2.1

[118] K. Solaimani, S. Modallaldoust, and S. Lotfi. Investigation of land use changes on soil erosion process using geographical information system. (3/603010), 2009. 5.1

[119] David Stavens and Sebastian Thrun. A self-supervised terrain roughness estimator for off-road autonomous driving. In *UAI*. AUAI Press, 2006. ISBN 0-9749039-2-2. 4.1

[120] A. Stentz. Robotic technologies for outdoor industrial vehicles. In *Proceedings of SPIE AeroSense*, 2001. 1.1

[121] Anthony Stentz. Optimal and efficient path planning for partially-known environments. In *ICRA*, pages 3310–3317, 1994. 3.3.1

[122] Anthony Stentz, Cristian Dima, Carl Wellington, Herman Herman, and David Stager. A system for semi-autonomous tractor operations. *Autonomous Robots*, 13(1):87–104, 2002. 1.1

[123] Anthony Stentz, Alonzo Kelly, Peter Rander, Herman Herman, and Omead Amidi. Real-time, multi-perspective perception for unmanned ground vehicles. In *Proceedings of the Association for Unmanned Vehicle Systems International*, 2003. 4.1

[124] Anthony Stentz, John Bares, Thomas Pilarski, and David Stager. The crusher system for autonomous navigation. In *AUVSIs Unmanned Systems North America*, August 2007. 3.3

[125] Matthew J. Streeter and Daniel Golovin. An online algorithm for maximizing submodular functions. In *NIPS*, pages 1577–1584. MIT Press, 2008. (document), B.5, B.5, 7, B.5

[126] GT Sung and IS Gill. Robotic laparoscopic surgery: a comparison of the da vinci and zeus systems. *Urology*, 58(6):893–8, 2001. 6.1

[127] L. Tarassenko, P. Hayton, N. Cerneaz, and M. Brady. Novelty detection for the identification of masses in mammograms. In *Proceedings of the Fourth International IEEE Conference on Artificial Neural Networks*, volume 409, pages 442–447, 1995. 5.1

[128] David M. J. Tax and Robert P. W. Duin. Support vector domain description. *Pattern Recognition Letters*, 20(11-13):1191–1199, 1999. 5.1

[129] Sebastian Thrun, Michael Montemerlo, Hendrik Dahlkamp, et al. Stanley: The robot that won the darpa grand challenge. *Journal of Field Robotics*, 23(9):661–692, June 2006. 1, 1.1

[130] Michael E. Tipping. Bayesian inference: An introduction to principles and practice in machinelearning. In *Proceedings of the Advanced Lectures on Machine Learning*, 2003. 4.2.2

[131] C. Tomlin, G.J. Pappas, and S. Sastry. Conflict resolution for air traffic management: a study in multiagent hybrid systems. *Automatic Control, IEEE Transactions on*, 43(4): 509–521, Apr 1998. ISSN 0018-9286. 6.1

[132] Paul Tompkins. *Mission-Directed Path Planning for Planetary Rover Exploration*. PhD thesis, Robotics Institute, Carnegie Mellon University, 2004. 4.1

[133] C. Tottrup. Forest and land cover mapping in a tropical highland region. *Photogrammetric Engineering and Remote Sensing*, 73(9):1057–1066, September 2007. 5.1

[134] Chris Urmson. *Navigation Regimes for Off-Road Autonomy*. PhD thesis, Robotics Institute, Carnegie Mellon University, 2004. 4.1

[135] Chris Urmson, Joshua Anhalt, Drew Bagnell, et al. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466, 2008. ISSN 1556-4959. 1, 1.1

[136] A. Vahidi and A. Eskandarian. Research advances in intelligent collision avoidance and adaptive cruise control. *Intelligent Transportation Systems, IEEE Transactions on*, 4(3): 143–153, 2003. 6.1

[137] Nicolas Vandapel, Raghavendra Rao Donamukkala, and Martial Hebert. Experimental results in using aerial ladar data for mobile robot navigation. In *Proceedings of the International Conference on Field and Service Robotics*, 2003. 4.1

[138] Paul Vernaza, Ben Taskar, and Daniel D. Lee. Online, self-supervised terrain classification via discriminatively trained submodular markov random fields. In *ICRA*, pages 2750–2757. IEEE, 2008. 4.1

[139] M. Wada, Kang Sup Yoon, and H. Hashimoto. Development of advanced parking assistance system. *Industrial Electronics, IEEE Transactions on*, 50(1):4–17, Feb 2003. ISSN 0278-0046. 6.1

[140] C.C. Wang, SR Kulkarni, and HV Poor. Bandit problems with side observations. *IEEE Transactions on Automatic Control*, 50(3):338–355, 2005. 6.2.1

[141] C.J.C.H. Watkins. *Learning from delayed rewards*. Cambridge University, 1989. 6.2.2

[142] M.L. Weitzman. Optimal search for the best alternative. *Econometrica: Journal of the Econometric Society*, pages 641–654, 1979. 6.2.1

[143] Carl Wellington. *Learning a Terrain Model for Autonomous Navigation in Rough Terrain*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, December 2005. 3.3.1, 4.1

[144] Carl Wellington, Aaron Courville, and Anthony Stentz. Interacting markov random fields for simultaneous terrain modeling and obstacle detection. In *Proc. of Robotics Science and Systems*, June 2005. 5.4.4

[145] K. Worden. Structural fault detection using a novelty measure. *Journal of Sound and Vibration*, 201(1):85–101, 1997. 5.1

[146] B.M. Yamauchi. Packbot: a versatile platform for military robotics. In *Proceedings of SPIE*, volume 5422, pages 228–237, 2004. 6.1

[147] Holly A. Yanco and Jill L. Drury. Rescuing interfaces: A multi-year study of human-robot interaction at the AAAI robot rescue competition. *Auton. Robots*, 22(4):333–352, 2007. 6.1

[148] R. Zabih, O. Veksler, and Y. Y. Boykov. Fast approximate energy minimization via graph cuts. In *ICCV*, pages 377–384, 1999. 5.4.4