ACTIVE LEARNING FOR OUTDOOR PERCEPTION

Cristian Dima

CMU-RI-TR-06-28

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

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

May 2006

ABSTRACT

Many current state-of-the-art outdoor robots have perception systems that are primarily hand-tuned, which makes them difficult to adapt to new tasks and environments.

Machine learning offers a practical solution to this problem. Assuming that training data describing the desired output of the system is available, many supervised learning algorithms exist for automatically adjusting the parameters of possibly complex perception systems. This approach has been successfully demonstrated in many areas, and is gaining momentum in the field of robotic perception.

An important difficulty in using machine learning techniques for large scale robotics problems comes from the fact that most algorithms require labeled data for training. Large data sets occur naturally in outdoor robotics applications, and labeling is most often an expensive process. This makes the direct application of learning techniques to realistic perception problems in our domain impractical.

This thesis proposes to address the data labeling problem by analyzing the unlabeled data and automatically selecting for labeling only those examples that are important for the classification problem of interest. We present solutions for adapting several active learning techniques to the specific constraints that characterize outdoor perception, such as the need to learn from data sets with severely unbalanced class priors. We demonstrate that our solutions result in significant reductions in the amount of data labeling required by presenting results from a large amount of experiments performed using real-world data.

ACKNOWLEDGMENTS

THIS thesis would have never come into existence without the support I have received from my advisors, Martial and Tony. Not only did they provide invaluable guidance on research related issues, but their passion for research and professorship will continue to be a model for me far beyond my years in graduate school. I would also like to thank the other members of my thesis committee, Jeff Schneider and Larry Matthies for the time they took to review my research and for their insightful comments on my dissertation.

My colleague and friend Carl Wellington provided me with a tremendous amount of support over the last six years we have spent working on the Deere project. We have shared joys, sorrows and quite a few all-nighters, but his openness and healthy laughter made it all more enjoyable.

I would not have enjoyed my graduate years in Pittsburgh as much had it not been for my friends Mihai and Raluca Budiu. Their passion for classical music and great movies, together with all the concerts, picnics and stroller-pushing sessions that we shared made my social life infinitely better.

My sons, Simon and Adam, and my wife Nathalie contributed to this thesis more than one could ever imagine. Their smiles give my life a meaning and put everything in perspective.

Finally, I would like to dedicate this thesis to my beloved parents, Gabriel and Veronica, to whom I owe it all.

This document was prepared through collaborative participation in the Robotics Consortium sponsored by the U. S. Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement DAAD19-01-2-0012. The U. S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon. The development of the autonomous vehicle and the infrastructure used for most of the experimentation described in this document was generously funded by John Deere & Co. under the Vehicle Safeguarding Project (contract number 476169) at Carnegie Mellon University.

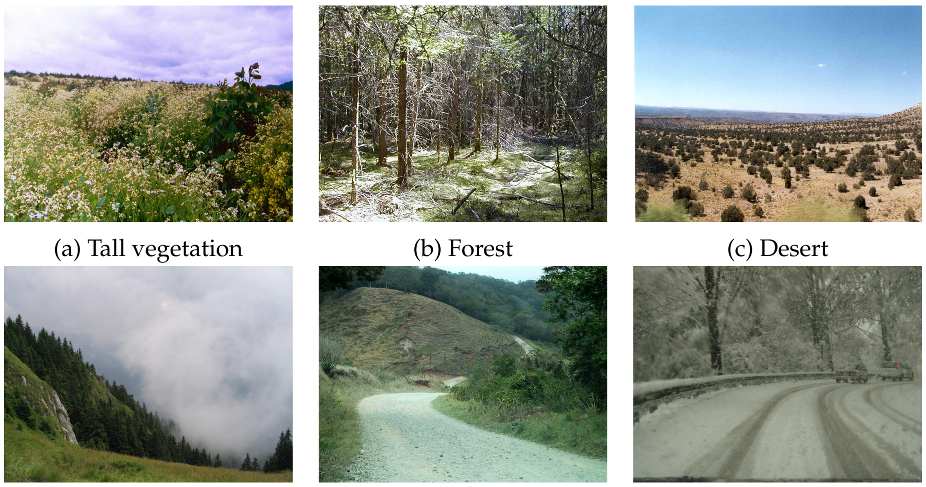
TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

Introduction

THE problem that motivates this thesis is one of great importance to the mobile robotics field: the need for reliable perception for off-road navigation. We are interested in enabling autonomous vehicles to ”see” obstacles and classify terrain, so that they could navigate safely through environments such as forests, fields and mountains. In these types of environments, any location where one would not want a vehicle to drive is an obstacle: trees, rocks, trenches, wires, poles, deep water and dense vegetation are just a few examples.



(a) Tall vegetation (b) Forest (c) Desert

(d) Mountains (e) Dirt road (f) Winter conditions

FIGURE 1.1. Different types of environments in which a robot could be expected to navigate reliably. The dramatic range of conditions make it very difficult to develop general perception systems.

Many of the current state-of-the-art robotic systems have demonstrated satisfactory performance in challenging types of terrain. Unfortunately, the solutions proposed are not general. In order to achieve good performance, these systems use specialized perception algorithms that are often manually tuned to a specific environment. They exploit structure and constraints that are valid locally, but do not necessarily apply to other environments. If such a robotic system is forced to function in a new type of terrain (a few possible examples are presented in Figure 1.1), performance will drop dramatically, because many of the assumptions the system relies on become unrealistic. The same drop in performance would occur if one tried to use the perception algorithms on robots of different scales, or equipped with slightly different sensors.

Exploiting local structure for improving perception capabilities is *not* a problem. One could argue that the reason humans can drive so well on challenging terrain is not superior sensing, but rather our ability to make very good assumptions about the environment based on domain specific knowledge. When driving through a forest with tall vegetation, we use different rules for detecting fallen trees then when going through the desert. Our strength comes from the fact that we can adapt these rules without needing manual tuning, thanks to our capacity to learn.

The main limitation of current systems is that they are *not easily adaptable*. In order to achieve good performance in new situations, hand-tuned systems require additional tuning by human experts. Since current state-of-the-art systems tend to be complex, the re-tuning process is slow and expensive, which makes this approach inappropriate for applications requiring frequent reconfigurations of the perception systems. The key application domains for outdoor mobile robotics require frequent reconfigurations.

To make outdoor perception easily adaptable, we will need to move away from manual tuning and rely – at least partially – on automated statistical learning algorithms. A simple example is an obstacle detection system based on a neural network that uses sensor readings converted to *features* along with ”obstacle/non-obstacle” *labels* provided by a human expert to learn about the “obstacle” concept.

When deployed, the system makes obstacle predictions based on the features extracted from the data coming from the sensors on the vehicle (color cameras, laser radars, infrared cameras, radars) and the model produced by the learning algorithm.

Algorithms that have statistical learning techniques at their core have quickly come to dominate fields such as computer vision, automatic target recognition and speech processing. In certain cases, it is actually hard to think of any manually derived algorithms. For example, trying to produce classification rules for distinguishing between “grass”,“trees”,“soil’ and “rocks” based on texture information is a hopelessly difficult task, since texture features are typically of very high dimensionality. It is unrealistic to expect a human expert to produce the hundreds of thresholds and combination rules required. It makes a lot more sense to use a generic classifier such as a neural network or a support vector machine to accomplish this task, since such algorithms can use high dimensional feature vectors much more effectively.

A frequent argument against using automated learning techniques for challenging perception problems is that they necessarily have to learn everything from the data, much like neural networks or decision trees do. While this has been the case with most early robotic systems using learning, several successful applications of Bayes networks or other learning methods based on graphical models have been demonstrated in other domains, indicating that learning algorithms can incorporate and benefit greatly from human expert knowledge of the problem of interest.

Human expertise can be combined with machine learning approaches either directly or indirectly. In our domain, the direct approach could consist of combining the outputs of hand-derived classifiers developed by human experts with those of general learning algorithms. One could even use training data to learn how to combine such classifiers, as we have demonstrated in [**19**].

The indirect approach is based on the observation that humans can often produce good perception algorithms, but they cannot easily “optimize” parameters manually. Manually tuning an outdoor perception system involves a lot of trial and error: we actually perform a manual heuristic search through the parameter space, using an error metric based on how much the behavior of the system agrees with the desired behavior.

There are good reasons to expect that if this error metric is clearly defined in a functional form and the "desired behavior" is translated into a labeled dataset, a computer could explore a much larger part of the parameter space. If the human expert can provide good starting values or some reasonable ranges for most of the parameters, various optimization algorithms can be used to find very good locally optimal parameter values.

We thus argue that machine learning is a crucial ingredient for successful outdoor perception systems: it makes them easy to tune, it enables the use of high-dimensional input data that could potentially contain more information, and it provides methods for gracefully combining expert knowledge and computational power. There is however an important problem that we have not addressed yet: obtaining the training data required by most machine learning algorithms.

The need for labeled data is a well-know problem affecting all supervised learning applications that require large amounts of training data and for which labeling is a relatively expensive process. Obstacle detection or terrain classification in unstructured environments have both these properties. A learning-based system cannot be expected to detect a certain type of obstacle unless is has been previously exposed to data from a similar distribution. As a result, a large amount of labeled data is needed in order to detect a large variety of obstacles. The need to observe the obstacles under different environmental conditions that might affect their signature in the feature space[1] only makes the problem worse. The same is true for terrain classification.

Figure 1.2 describes the typical steps that need to be taken in order to generate a labeled dataset for obstacle detection. Of those, the first one turns out to be the least demanding: a human can easily tele-operate a mobile robot, or a data logging

[1]For example at different times of the day for illumination and temperature changes.

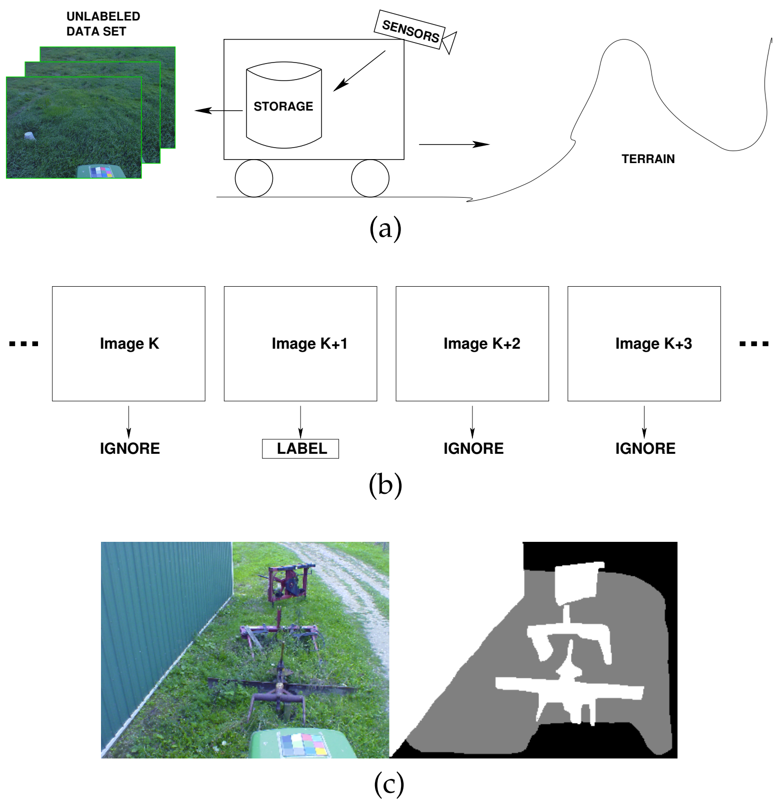


FIGURE 1.2. The steps involved in generating a labeled dataset in a "traditional" fashion: (a) Recording a data log, (b) Selecting the scenes to be labeled, (c) A raw image and the corresponding label map provided by a human expert. In a typical supervised learning application in robotics, all three steps require human assistance.

device can be mounted on a non-robotic vehicle that performs a task that is representative for the application of interest. Large amounts of *unlabeled* data can be recorded with relatively low effort, and without requiring a high level of expertise from the human involved.

The most difficult problem consists in selecting those scenes from the unlabeled data set that will be presented to the human expert. Labeling data exhaustively is close to impossible for even moderately sized data sets, which means that a much smaller subset needs to be extracted from the unlabeled data set. The choice of scenes to be labeled is directly linked to the generalization performance obtained by the learning perception system after training. An expert choosing the data to be labeled would have to be able to reason about the data patterns contained in each

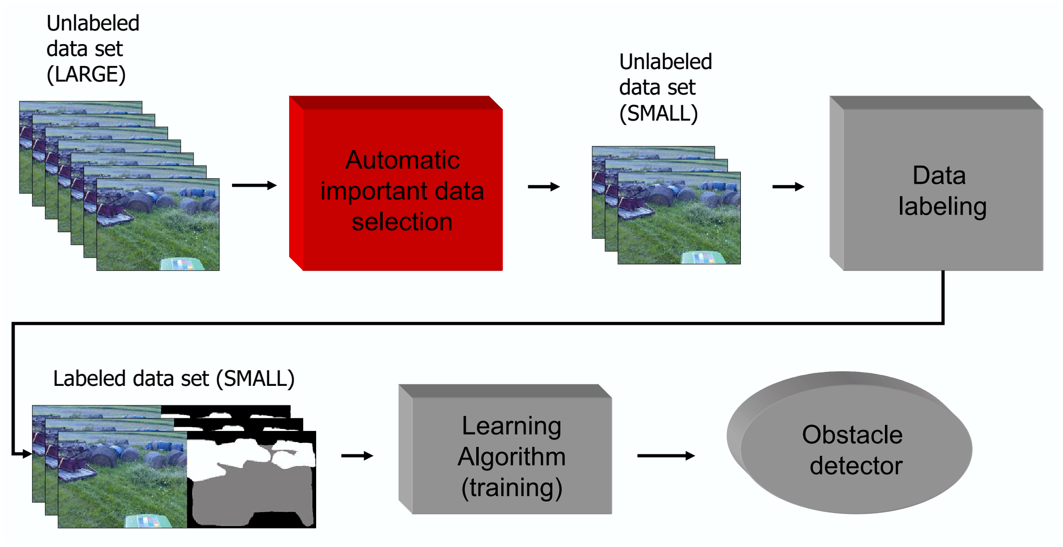


FIGURE 1.3. A high level representation of the focus of this thesis. We present methods for automatically identifying the important data to label (the red box). A small number of hopefully informative images are extracted from a large data set. As a result, the labeling effort required to train a classifier is reduced.

scene and their implications on the performance of the learning algorithm. This is a difficult task requiring special skills, especially when we consider that for large data sets one could have fractions of a second to decide if a scene is labeled or not.

As we indicate in Figure 1.3, solving the data labeling problem is the main topic of this thesis. We describe several approaches for automatically selecting important images for labeling, and demonstrate that they significantly reduce the effort required in order to learn from large, realistic data sets. The solutions we propose reduce not only the amount of time required for labeling, but also the level of expertise expected from the human labeling data. Together, these two effects have the important consequence of making the use of machine learning techniques feasible for several applications in outdoor perception for robotics applications.

In the remaining part of this chapter, we will provide an overview of the related work and provide a more detailed list of the contributions resulting from this thesis.

1. Related work

We believe this thesis is a good illustration of the type of interdisciplinary efforts that are often required in robotics. Our research of course relates to much of the previous work in robotic perception, but also to machine learning and in particular to the active learning and data mining research areas.

For the very beginnings of mobile robotics, perception has been identified as a key problem that needs to be addressed before any truly autonomous navigation is possible. Early on, researchers identified the need for multiple sensing modalities, and designed hand-tuned systems for obstacle detection. In 1979 the robot HILARE [**16**] was designed to use vision, acoustic sensors and a laser range-finder for operating in unknown, man-made environments. Around 1983 mobile robotics moved outdoors, first with Moravec's Stanford Cart and CMU Rover [**57**] and then with both the Ground Surveillance Robot (GSR) [**33, 34**]) and DARPA Autonomous Land Vehicle (ALV) [**13, 14, 58**] navigating in unknown natural terrain. The CMU Rover used sonars, TV cameras and infrared sensors together with contact switches. The GSR and ALV used color vision, sonars and laser range-finders.

A long series of 11 autonomous vehicles resulted from the CMU NAVLAB program [**82, 30**] initiated in 1984 and still going on today. The NAVLAB effort led to a number of innovations, but the most important for this thesis is the fact that Pomerleau [**67, 68**] demonstrated the first successful application of machine learning methods to the problem of mobile robot navigation with the ALVINN system. The author demonstrated that a neural network taking as input low resolution images of a road can be trained to generate as output the steering angles necessary for road following. Obtaining sufficient training data for the neural network was identified as a major problem, and the author proposed to artificially alter input-output pairs in order to increase the number of training patterns available. This technique is still in use today in many state-of-the-art computer vision systems for face and digit recognition. A few years later, Davis and Stentz [**15**] proposed the MAMMOTH system which employed a neural network to learn how to combine steering angles produced by other neural networks using image and laser data.

Interestingly, after the early learning success with ALVINN, Pomerleau developed an even better system, RALPH [**69**], that was entirely hand-coded and was inspired by observing the features that ALVINN identified as important for road following. For a while, robotic perception seemed to revert to hand-tuned systems.

Non-learning systems such as the ones described in [**84, 63, 92**] have used rule-bases or different behaviors to demonstrate reasonable performance in quite challenging environments. Other researchers [**74, 78**] took hand-tuning to the extreme, developing rules that try to account for all types of terrain and as many environmental conditions as possible, such as wind speed, humidity, temperature, illumination, time of the day or season. The degree of success achievable with this type of approach is still to be determined.

We believe the most remarkable non-learning systems for outdoor perception have been proposed by researchers at JPL who, with a few notable exceptions such as [**46, 6**], have focused on hand-derived specialized detectors for challenging obstacles. Under the DEMO III [**83**], PerceptOR [**25, 23, 24**] and CTA projects funded by DARPA and ARL, their group developed algorithms (mostly vision based) for detecting positive and negative obstacles, tree trunks, water bodies, overhang and excessive slope hazards [**85, 48, 39, 71**]. Since the algorithms they propose tend to have a certain number of hand-tuned parameters, they are a great example of what we refer to in the introduction as specialized detectors encapsulating a great amount of human insight, and that could benefit from the capability to automatically tune parameters. A nice overview of the recent JPL work on obstacle detection is provided in [**70**].

At the same time, the difficulties encountered in manually tuning hand-derived algorithms led to a renewed interest in learning based techniques. Some researchers have used Gaussians [**40**] and radial basis function networks [**75**] for road following, while others have used 2-D HMMs to filter pixel classification in image streams [**77**]. Rosemblum [**73**] proposed a navigation system based on an "artificial immune system" (AIS)[2] to learn the types of terrain that are safe to traverse

[2]Extremely similar to reinforcement learning

based on experience or example learning. Witus [**90**] confirmed that a neural network using features extracted from images can successfully predict the roughness and traversability of surfaces in front of a robot. Vandapel and Hebert [**37, 87**] proposed algorithms that combine the extraction of specialized features from 3-D point cloud statistics with learning-based modeling of probability densities, in order to discriminate between vegetation, linear and planar surfaces. The group of researchers at NIST, whose experimental setup happens to be closest to the one we are using, have presented results in road detection [**72, 38**] obtained by training neural networks using laser, color and texture features. Towards the end of the PerceptOR program, Ollis [**62**] successfully demonstrated a system using Bayesian learning for off-road obstacle detection.

The increased interest in learning based algorithms has been confirmed and supported by the creation of a new DARPA program focused on learning for outdoor perception. Launched in 2004, the LAGR[3] program has succeeded in attracting top researchers in machine learning, computer vision and mobile robotics to focus on the difficult problems posed by outdoor perception, and has already demonstrated its great potential to lead to important advances in our field.

The work presented in this thesis is related quite directly to most of the learning techniques mentioned above, in that it addresses the important problem of data labeling. To develop practical algorithms for reducing the amount labeled data required for training supervised learning algorithms, we have adapted techniques from both the active learning and the data mining areas to the specific constraints of our domain. Since more context would be beneficial for understanding the relation between our research and previous work in active learning and data mining, we will only briefly describe them here, postponing a more in-depth discussion until chapters 2 and 3 respectively.

Active learning is an area of machine learning that addresses the situation in which a large amount of unlabeled data is available, and only a limited subset of it can be labeled. Active learning algorithms reason about the distribution of unlabeled data, and have the freedom of choosing the order in which data points

[3]Learning Applied to Ground Robots

get labeled. The goal of active learning is to obtain as good as classifier as possible given the amount of labeling that is available. Applications of this approach have been successfully demonstrated, mostly in the text and web page classification area **[45, 1, 59, 86, 76]**.

A different flavor of active learning (which is closer to the DOE[4] area of statistics) is concerned with maximizing an unknown and usually noisy function in the case where estimating the function for different settings of the inputs is expensive **[55, 56, 80]**. This is an extremely important problem that appears naturally in marketing and in pharmaceutical applications. A somewhat related application domain of active learning is robot control, where active learning techniques have been used for having manipulators learn tasks such as bouncing a ball or throwing a ball at a basket **[53, 52]**. While not directly related to our task of interest which is classification, these are still important examples of domains in which the application of active learning leads to significant reductions in the amount of function value evaluations required.

This thesis does not propose new active learning algorithms, nor does it attempt to identify the best active learning algorithm for our domain. Instead, our main goal is to explore the applicability of the active learning approach to our domain, and to verify that reductions in data labeling can be achieved. Since standard active learning methods are not directly applicable to our problem of interest, a couple of modifications described in more detail in Chapter 3 are proposed.

It is one of the constraints on the direct application of active learning techniques to outdoor perception that connects our research to the data mining field, and in particular to work in anomaly detection. As we will show in the following chapters, our problem is characterized by severely unbalanced class priors, which cause problems with the initialization step required by most active learning methods. The solution we propose to this problem, which is to explore regions of the feature space that are sparsely populated, is also frequently used for detecting anomalies in data sets, although for very different purposes. As its name suggests, anomaly detection is the problem of identifying "out of the ordinary" data points

[4]Design Of Experiments

(which could be events, measurements, etc.) in data sets. There is large variety of highly practical applications of anomaly detection, including computer security and intrusion detection [**44**], fraud detection, disease outbreak detection [**91**], data cleaning [**21**], identifying regime changes in time series [**5**] and background subtraction for video analysis [**66**]. Although we will elaborate more on this topic in Chapter 3, an extremely concise description of the relation between our work and the anomaly detection area could be the following: researchers in anomaly detection often use machine learning for anomaly detection, while we exploit techniques from anomaly detection for machine learning[5].

Having presented most of the previous work that relates to our research, we dedicate the next section to stating the main contributions of this thesis.

2. Thesis Contributions

This thesis focuses on the problem of making supervised learning techniques practical for applications to large-scale perception problems such as the ones occuring in outdoor robotics applications. Its main contributions are:

- We demonstrate that active learning techniques can be adapted to the constraints imposed by the outdoor robotics domain, and they result in significant reductions in the amount of data labeling required by supervised learning algorithms. To the best of our knowledge, this is the first application of active learning to robotic perception.
- We propose the Unlabeled Data Filtering algorithm, based on detecting unusual patterns in unlabeled data, and show that it can be effectively used both as a stand-alone active learning algorithm and as an initialization method for standard active learning methods.

The effectiveness and practicality of all the solutions proposed in this thesis are demonstrated through experiments based on several real world data sets.

[5]More precisely for active learning

CHAPTER 2

A Standard View of Active Learning

1. The Challenges of Learning from Large Data Sets

THE previous chapter presents some of the benefits that can be expected from applying learning-based approaches to the domain of outdoor perception. Learning leads to more powerful, better performing classifiers that require less manual tuning. We have argued that the reduced amount of tuning required opens new possibilities by enabling robotic systems to quickly be adapted to exploit local terrain characteristics.

There is however an important caveat: the methods discussed so far are instances of *supervised learning*. As a result, they can only be adapted as fast as one can provide representative labeled data to learn from. Since the classification problems in our domain often require learning from very large data sets, being able to *efficiently* obtain labeled data is of utmost importance. Fortunately, several reasonable solutions exist for alleviating data labeling requirements.

For certain learning problems, labels can be obtained at very low cost. In **[89]**, the authors describe a system for predicting the height of the load bearing surface from forward-looking sensor data. The system is trained by simply driving the vehicle over the terrain of interest. In this specific application, ground truth data about the true height of the terrain can be easily obtained: the sensors make measurements about the terrain the vehicle is about to drive over, and the position of the vehicle's rear wheels is recorded as the vehicle drives over the terrain (see

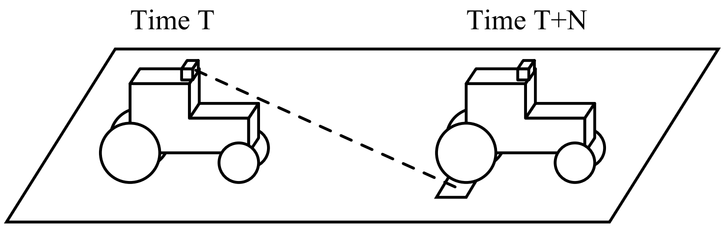


FIGURE 2.1. Obtaining labeled data for ground height estimation. At time T, measurements about a specific location are taken and converted to a feature vector x. At time $T+N$, the vehicle drives over that location, and a measurement of the ground height y is obtained. The learning problem consists in finding a function F such that $F(x) \simeq y$. Reproduced from [**89**] with the author's permission.

Figure 2.1). The training process consists in fitting a regression model F that approximates the measured ground heights y_i where $i = 1 \ldots k$ with $F(x_i)$, where x_i is the feature vector based on the sensor measurements made about the region with height y_i. Precisely estimating the position of the vehicle is crucial for both estimating the positions of the rear wheels and establishing the correspondence between the measurements x_i taken with the forward-looking sensors and the height estimates y_i obtained with the wheels. The practicality of this approach comes from the fact that for the regression task of interest, both the inputs and the desired outputs of the system can be obtained at very low cost.

The idea of applying the same labeling process to the obstacle detection domain is very tempting, but unfortunately impractical. Automatically obtaining labels for obstacle examples generally requires a robot to interact with an obstacle, detect this interaction and then use that information for labeling. As a result, the only cases in which a robot can autonomously label obstacle data are the ones that involve a small robot traveling at low speeds in benign environments, so that the collisions required for labeling obstacle data are tolerable.

The previous approach fails because it is not always possible to easily obtain obstacle labels. It seems natural to wonder if in certain cases one could make certain assumptions and avoid labeling obstacles altogether, only building a model of the "safe to traverse" class.

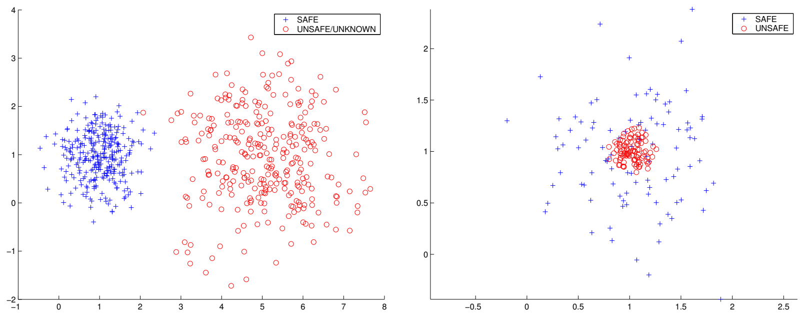


FIGURE 2.2. Making classifications after only modeling one class is not a generally applicable solution. LEFT: A data distribution on which setting a threshold on the likelihood of the data under the "SAFE"class (blue crosses) could work well. RIGHT: A data distribution on which the method fails. The mean of the very compact "UNKNOWN" class overlaps with the mean of the high-variance "SAFE" class.

Instead of the typical "obstacle/non-obstacle" classification problem, let us think about trying to distinguish between "safe" terrain –such as the one that a robot is driven over under human control– and "unknown" terrain, which has a feature signatures that are different from what has been observed as "safe". In probabilistic terms, if we observe the feature vector x, we are interested in obtaining $P(\text{safe}|x)/P(\text{unknown}|x)$, which can be written as

$$\frac{P(\text{safe}|x)}{P(\text{unknown}|x)} = \frac{P(x|\text{safe})P(\text{safe})}{P(x|\text{unknown})P(\text{unknown})} \tag{2.1}$$

If we are willing to make the strong assumption that observing any x is equally likely to appear in the "unknown" class, we obtain that $P(\text{safe}|x)$ is proportional to $P(x|\text{safe})$. This suggests we could model the probability density function of the "safe" class, $P(x|\text{safe})$, and simply set a threshold on its value so that patterns that have lower likelihood are considered to be "unknown". Although the assumption we have made is that $P(x|\text{unknown})$ is uniform, in reality this approach will work well anytime the function $P(x|\text{unknown})$ will not have strong peaks in the region where $P(x|\text{safe})$ is higher than the threshold.

Despite these strong assumptions, modeling only the "safe" class can be a good approach for certain obstacle detection scenarios, especially in highly structured environments such as the ones encountered in agricultural applications, where the

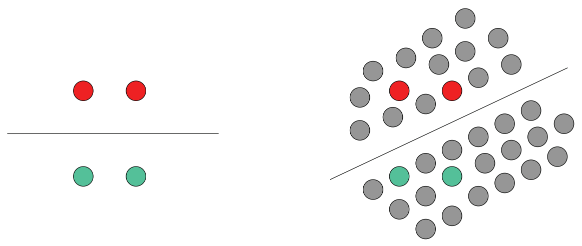


FIGURE 2.3. Learning with labeled and unlabeled data. (LEFT) After observing only the four labeled data points, the most appropriate decision boundary would be a horizontal line. (RIGHT) If additional unlabeled data (grey points) is taken into account, a different decision boundary seems more likely.

class density $P(x|\text{safe})$ is very compact. As we show in Figure 2.2, there are many other cases in which this approach would fail.

A well researched method for reducing labeled data requirements is to learn from both labeled and unlabeled data. Known as *semi-supervised learning*[**29**], this method has been successfully demonstrated in several text and web page classification applications[**61, 42, 9**]. The intuition behind it is simple: having a large amount of unlabeled data can in theory provide valuable information about the distribution of the data.

In Figure 2.3, we present a case in which observing the distribution of the unlabeled data in addition to the labeled data would suggest a dramatic change of the decision boundary. If we consider the geometric distance between the points in Figure 2.3 to be proportional to their dissimilarity, the location of the new decision boundary seems obvious to the human eye: the two clusters of points on each side of the boundary are compact, meaning that the points in each cluster are likely to be very similar to each other, and thus have similar labels. Furthermore, the decision boundary does not separate any two points that are very close to each other, which suggests that it corresponds to a natural separation in the data.

Unfortunately, in real world classification examples the problem is more difficult than presented in our figure. The problem of finding decision boundaries that minimize the separation of similar data points has been successfully addressed

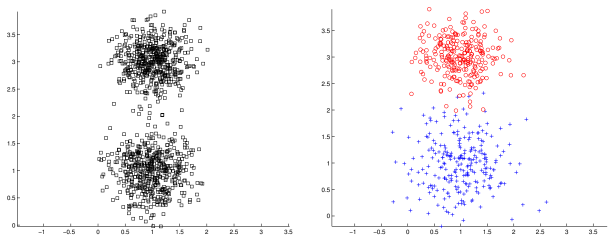


FIGURE 2.4. Reducing the labeling requirements by random sub-sampling: the data in the plot on the right was obtained by randomly sub-sampling half the unlabeled data from the plot on the left, and labeling it. If a model is learned using the labeled data in the right plot, we could hope the model is similar to the one obtained if all the data in the left plot was labeled.

[8, 42], but in general it is still challenging to find good distance metrics that results in compact clusters and good inter-cluster separation. Especially for learning problems that can potentially have a high degree of class overlap, semi-supervised learning can easily diverge to undesired classification boundaries.

Going back to the beginning of our argument about different methods for labeling data for learning from large data sets, one would notice that we have not discussed one of the simplest solutions: *random sub-sampling*. If it is impossible to label all the unlabeled data but we still want to learn from a distribution that is close to the original one, we could uniformly sample from the unlabeled data to obtain a data set that is smaller but hopefully representative. When a human expert labels the reduced set of unlabeled data and a model is fit using the labels, we can hope that the model will be similar to the one we would obtain if we had labels for all the unlabeled data. A representation of this situation is presented in Figure 2.4.

There are two disadvantages to reducing the amount of data to be labeled by random sub-sampling. The first, which is less severe, relates to the fact that when performing random sub-sampling we are equally likely to discard any data point, regardless of its position in the feature space with respect to the decision boundary. As we show in Figure 2.5, we would prefer to discard more of the points that are far from the decision boundary and retain (and label) as many as possible in the decision region.

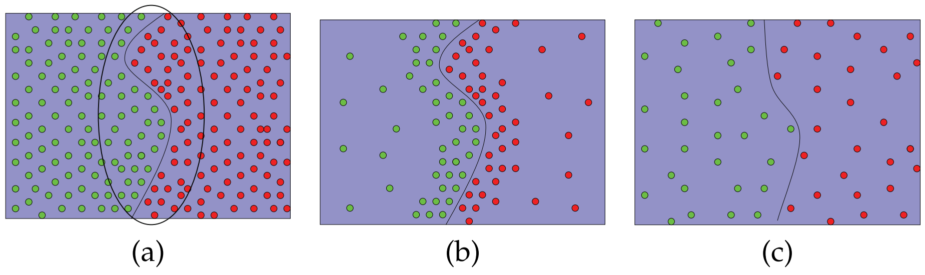


FIGURE 2.5. When reducing the size of a large data set (a) by discarding some of the data points, we would prefer to discard more of the points that are far from the decision boundary (b). Random sub-sampling is equally likely to discard any data point, so the data distribution we would obtain is the one represented in (c).

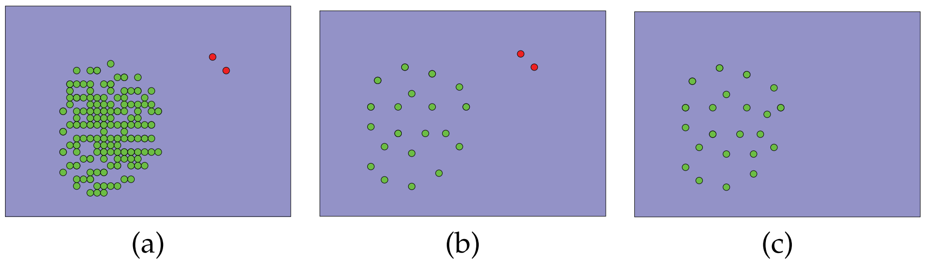


FIGURE 2.6. Applying random sub-sampling to very unbalanced data sets can have serious consequences. In (a) we depict a data set containing many "non-obstacle" examples (green points) and very few obstacles (red points). Since retaining the "obstacle" examples in the data set to learn from is crucial, the reduced data set we are interested in obtaining is similar to the one depicted in (b). Unfortunately, it is very likely that random sub-sampling will discard most of the examples from the infrequent class (c).

A more severe drawback of random sub-sampling occurs in the cases in which the class frequencies in the unlabeled data set are extremely unbalanced. Such data distributions occur quite naturally in the obstacle detection domain, since in many cases, a data set recorded during normal operation is likely to contain a far fewer examples from the "obstacle" class than from the "non-obstacle" class. A typical example would be a data set collected by a vehicle operating on a golf course, where thousands of redundant images of grass would be recorded before observing an obstacle.

In such cases, applying random sub-sampling is likely to result in the situation depicted in Figure 2.6, which can be catastrophic in an obstacle detection application.

The solution we propose for the data labeling problem can be succinctly described as a set of algorithms designed to address the random sub-sampling backdraws described in Figures 2.5 and 2.6. In the remainder of this chapter we will introduce a set of known techniques called *active learning*, designed to identify unlabeled data points that are likely to have a strong influence on the learned decision boundary once labeled. A new algorithm which we designed to address the problem depicted in Figure 2.6, and which was found to be very effective both as a stand-alone data reduction technique and as an initialization method for standard active learning algorithms will be described in Section 1 of Chapter 3.

In the sections that follow, we will briefly present the theoretical foundations of active learning and describe some of the better known algorithms which we will consider in our experimental evaluation.

2. Active Learning: A Brief Introduction

Active learning is a term describing learning methods that assume that not all the available unlabeled data can be labeled, and use various criteria for making intelligent decisions about which specific examples should be labeled.

Active learning presents three main paradigms: *constructive query*, *query filtering* and *pool-based* approaches. In the constructive approach (see [**11**],[**3**]) the algorithms can choose where in the input feature space it is most beneficial to obtain a label, and usually do so by trying to minimize the variance of future predictions. In contrast, in the query filtering and pool-based approaches it is assumed that a (possibly infinite) set of unlabeled samples is available, and the algorithms can only choose which data points to query. While the filtering approach assumes that new examples arrive as a random stream and the decision whether to query or not is made for each point individually, in the pool-based methods the entire unlabeled data set can be analyzed before choosing the next query point. Since the pool-based methods can use more information, they are generally expected to lead to faster learning than the filtering approaches.

Even though in theory the constructive approach could lead to maximally informative queries, for domains where it is hard to develop a good model for the generative process of the data and labels are provided by human experts, it is best to use pool-based approaches. In [**4**], the authors attempted to use the constructive active learning for solving a character recognition problem. However, since the neural network they used did not correctly model the interactions between character image pixels, they found that their algorithm generated query images that did not look like any recognizable characters.

For the classification problems we come across in the obstacle detection area, it is unfortunately the case that the set of features extracted from the sensor data is far from capturing the complexities of the true data generation process. Even if we could identify a maximally informative query point in our feature space, we could not generate the corresponding input data that would allow a human expert to label the query point. As a result, the active learning scenario we are considering is *pool-based active learning*, meaning that query examples need to be selected from a finite pool of candidates.

Most active learning techniques are iterative, alternating between choosing a new query point based on some measure of interest (which can be classification uncertainty, disagreement among a group of classifiers, etc.) and training one or more classifiers on the new data set obtained after completing the query. In cases where the computational complexity of training the classifier(s) is high and a large number of queries is needed for the learning task of interest, it is common to select a series of query points in a single step, based on the same classifier parameters. This is suboptimal because if we select several query points at a time, we are effectively forcing the algorithm to delay analyzing the information obtained from the queries until the labels for the entire batch are provided. For certain active learning scenarios this suboptimal approach is the only feasible solution due to computational constraints. In Section 2, we will describe a slightly different constraint that occurs in the obstacle detection domain, consisting of the need to select a fixed *block* of data points in a single step.

3. Query-By-Committee

Although the use of queries for learning was first studied in the 1980s, we consider that the most influential result in this domain was the development of the Query-By-Committee (QBC) algorithm **[81, 27]** which inspired many of the algorithms we discuss in this paper. The QBC algorithm is based on the Bayesian model of *concept* learning **[36]**. A concept c is a mapping $c : X \rightarrow 0, 1$ defined over an instance space X, where $c(x) = 1$ if x represents an instance of the concept and 0 if it does not. A *concept class* C is a set of concepts. A good concept example would be the concept of "chairs", and a concept class could be the set of household objects. In our specific scenario, a human expert labeling data in the obstacle/non-obstacle classes would be defining the "obstacle" concept that we would like an algorithm to learn.

A learning algorithm for a certain concept c is expected to produce a hypothesis h, such that the probability that $h(x) \neq c(x)$, the probability of error, is small. Following the notation in **[27]**, we can denote a sequence of unlabeled data by $\vec{X} = \{x_1, x_2 \ldots\}$ and use $\langle \vec{X}, c(\vec{X}) \rangle = \{\langle x_1, c(x_1) \rangle, \langle x_2, c(x_2) \rangle \ldots\}$ to denote the sequence of labeled examples produced by applying the concept c to the instances in $\vec{X}$. If $\vec{X}_{1\ldots m}$ is used to denote the sequence of first m elements from $\vec{X}$, then we can define the *version space* **[51]** generated by the sequence of labeled examples $\langle \vec{X_{1\ldots m}}, c(\vec{X_{1\ldots m}}) \rangle$ as V_m, the set of all concepts in C that are consistent with the first m labeled examples presented. The version space is a representation of all the information contained in the examples observed by the learning algorithm: the smaller the version space, the more effective the set of labeled examples was at reducing the number of concepts in C that are consistent with the labeled data. Before seeing any examples, the version space V_0 is equal to C, the entire concept class.

If we step back for a moment and return to our example of trying to learn the concept of "chairs", we will notice that if we are presented with a training set comprising a glass, a knife and a refrigerator and we are told that they are *not* examples of our target concept we will know more about "chairs" than if we were presented with three examples of glasses. In the first case, the size of the version space *decreases more* than in the second case, because in addition to the "glasses" concept, the "knives" and "refrigerators" concepts are eliminated as well from the set of household objects (our concept class) that, a priori, could have been examples of the "chair" concept. The rate at which the size of the version space decreases is considered a good measure of the progress of the learning process.

In **[27]**, the authors consider a two-class problem and define the *instantaneous information gain* as $-\log \Pr_{\mathcal{P}}(V_i)/\Pr_{\mathcal{P}}(V_{i-1})$, where V_i is the version space corresponding to the first i labeled examples and $\mathcal{P}$ is the distribution over the concept class $\mathbf{C}$ from which the target concept is chosen[1]. $\Pr_{\mathcal{P}}(V_i)$ is the probability mass from the distribution $\mathcal{P}$ that is contained by the version space obtained after seeing i labeled examples[2].

To measure the information to be gained from obtaining the label of a specific unlabeled example, one can compute the expected instantaneous information gain with respect to the probability that each one of the two possible labels occurs given the current version space. In other words, the probabilities of the two labels (p and $1-p$) are estimated over the entire set of hypotheses that are consistent with the labeled data accumulated so far[3]. Through some simple manipulation, it is shown that the expected gain of an example x_i given the current version space V_{i-1} is given by the Shannon information content of a binary random variable whose probability of being 1 is p:

$$\mathcal{G}(x_i|V_{i-1}) = \mathcal{H}(p) = -p\log p - (1-p)\log(1-p)$$

Thus, the unlabeled example whose labeling will result in the largest decrease of the version space is the one whose distribution over its labels –according to the hypotheses in the current version space– has the highest entropy. The entropy is estimated by sampling hypotheses from the current version space according to $\mathcal{P}$. These hypotheses form a committee which predicts the label of unlabeled

[1]Unlike in the PAC model, the Bayesian model of concept learning assumes that $\mathcal{P}$ is known to the learner.
[2]This is a more qualified definition of the size of the version space.
[3]An important assumption made by the QBC algorithm is that the hypothesis space is a superset of the concept space.

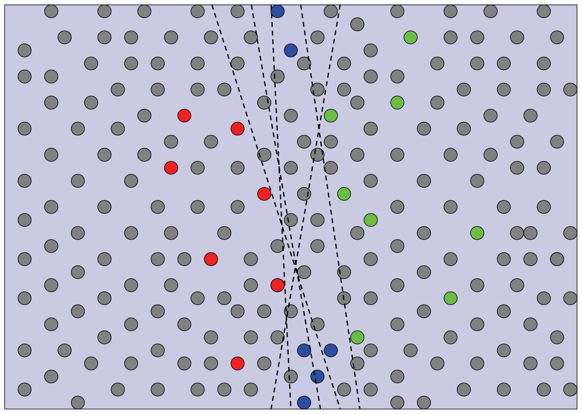


FIGURE 2.7. QBC is attempting to query points that are likely to result in a large decrease in the size of the version space. To estimate the information gain obtained from the future query, a set of hypotheses – represented as linear decision boundaries in the figure– are sampled from the version space determined by the already labeled points (in red and green). The unlabeled examples whose label is most uncertain given the current version space are represented in blue.

data points, and queries the unlabeled examples where there is disagreement in the committee. The process is depicted in Figure 2.7.

The authors point out that while the expected information gain is an attractive method for selecting queries, it is *not* sufficient for guaranteeing a large reduction in the expected prediction error. They are however able to prove that for certain classes of learning problems QBC does indeed guarantee an exponential decrease of the prediction error as a function of the number of queries. The proof is relatively involved, but at a high level it involves two steps: proving that having a lower bound on the information gain of the queries does guarantee a fast decrease in the prediction error, and then proving that for a restricted family of parameterized concept classes, the queries made by QBC have an expected information gain that is guaranteed to be higher than a constant.

Freund et al. [**27**] also make a few observations about a possible algorithm for minimizing the expected prediction error directly, instead of maximizing the expected information gain. They show that if two oracles **Sample** and **Gibbs** exist that can provide unlabeled data points and sample hypotheses from the version space according to $\mathcal{P}$, then one can select query points that will minimize the expected prediction error in more general situations. QBC is an efficient heuristic of this more general algorithm, which is shown to be much more intensive computationally. A few years later, Roy and McCallum [**76**] successfully addressed some of the computational issues and presented a working algorithm that uses Monte Carlo sampling to estimate the expected prediction error directly, using the same ideas as presented here. Their algorithm achieves important speedups by using incremental training and several rounds of sampling.

While the guarantees of QBC are interesting in themselves, the algorithm has few practical uses in the exact form in which it is presented. For most learning problems in the real world, the hypothesis space does not contain the target concept, and even if it did, it would often be impossible to find a hypothesis that is consistent with all the labeled data due to noise; as a result, the true version space could be empty. Sampling from the version space is also an issue. In specific cases (see McCallum and Nigam [**49**]) researchers use generative models and are able to sample from the distributions over their parameters to obtain members of the committee, but this is not a generally applicable solution.

In a slightly more general setting, one could use QBC with algorithms that are randomized and that will reach different hypotheses even when presented with the same training data (e.g. multilayer neural networks initialized randomly). For those who intend to use a deterministic algorithms (such as logistic regression) the solution is to apply one of the methods proposed by Abe and Mamitsuka [**1**]: Query-by-Bagging (QBBAG) and Query-by-Boosting (QBBOOST), which are discussed in the next section.

4. Query-By-Bagging

As its name suggests, the *Query-by-Bagging* (QBBAG,[**1**]) algorithm is a combination of the Query-by-Committee algorithm with the bagging algorithm introduced by Breiman in [**10**]. The combination is proposed as a method for making the QBC algorithm practical in the cases in which sampling from the true version space is not possible, and the target concept might actually not be contained in the hypothesis space.

Bagging is a learning method (applicable for both classification and regression) based on building a committee of learners on bootstrapped training data, and producing as output their averaged predictions. Bagging can be regarded as a variance reduction technique.

Query-by-Bagging starts from the implicit observation that even if the target concept is not contained in the hypothesis space and thus it is impossible to actually sample hypotheses from the true version space, we can still obtain a hypothesis that is in an *approximation* of the version space by training a learner on the available data. The question of how to sample from this approximate version space still remains valid, especially for deterministic learners. This is where bagging comes into play.

Since each random subsample used to train committee members is obtained by bootstrapping the original labeled data, all the committee members are trained on slightly different data sets that have similar distributions to the original labeled data set. As a result, the hypothesis produced by each member of the bagging committee after training can also be considered a sample from the approximation of the version space. These hypotheses can be used just as proposed in QBC, to identify unlabeled examples with high expected information gain by measuring the disagreement between the committee members. Since this time randomization is introduced in the training set, QBBAG can be used with both deterministic and stochastic classifiers.

In addition to Query-by-Bagging, Abe and Mamitsuka [**1**] also propose an algorithm called *Query-by-Boosting*, which a variant of QBC based on the boosting algorithm [**79**].

Boosting is a very popular learning technique, which, somewhat similarly to bagging, relies on combining a large number number of *weak* classifiers with relatively small learning capacity. The distinction from bagging comes from the fact that the members of the boosting committee are *not* trained using similar data distributions. Instead, boosting proceeds in an iterative fashion, and at each iteration the training examples either all get reweighted such that the misclassified examples get higher weights, or a new set of training examples for the next stage is obtained by sampling from a distribution that puts more probability mass over the misclassified examples. Intuitively, the algorithm can be described as a method for forcing the weak classifiers trained at different boosting stages to focus on classifying correctly the training examples which get misclassified by the previous stages. It has been demonstrated [**47**] that the training and reweighting scheme used in boosting is equivalent to performing gradient descent on an error function that exponentially penalizes misclassified examples, and that some other error functions less susceptible to overfitting might actually be more appropriate [**28**].

The connection that the authors of [**1**] make between boosting and QBC is not surprising, given their QBBAG approach: they propose to select for querying the unlabeled examples which have the smallest margin when classified by the boosting algorithm trained on all the labeled data available. These are the examples on which the weighted votes of the classifiers obtained from the different stages of boosting are most equally split. If we consider the different weak classifiers to be samples from an approximate version space, the examples with a small margin would then be the ones who are expected to lead to the highest information gain and thus reduce the version space at the fastest rate.

We believe that using boosting to sample from the version space is not a principled technique, because the members of the boosting committee are obtained by training weak classifiers on *different* distributions over the training data which would result in different version spaces. If we choose to look at boosting from a higher level and consider it as a classifier for which the split in the votes of the committee is an indication of the classification confidence, then Query-by-Boosting can be interpreted as an active learning method that queries the unlabeled data points with the highest classification uncertainty, as estimated based on a *single* hypothesis from the approximation of the version space. We will discuss more on this topic on the next section, in the context of uncertainty sampling.

In addition to the weak theoretical basis for Query-by-Boosting, it is known that the boosting algorithm has a tendency to overfit the training data when noise is present in the labels. This is very often true in our domain, which leads us to believe QBBOOST would not perform well. Furthermore, the experiments presented in [**1**] indicate that QBBAG and QBBOOST have similar performance in most cases, and QBBAG slightly outperforms QBBOOST in others. As a result, we have chosen not to include Query-by-Boosting in the experiments presented in this thesis.

An important choice for all active learning algorithms that use classifier committees to estimate the expected reduction in the size of the version space is the metric used for measuring disagreement in the committee. In the seminal QBC paper [**81**], the authors only consider committees of two classifiers, and exploit the infinite data stream model they use to obtain a simple rule for making queries: the algorithm asks for a label whenever the two classifiers in the committee disagree.

For cases in which pool-based active learning is performed, such as in the QBBAG algorithm, an extension of the previous idea is to select for labeling the example on which the votes of the classifier committee are most evenly split [**1**]. A more general solution was proposed by Argamon and Dagan [**2**] for the case in which the members of the classifier committee might have multi-class outputs. For committees consisting of k classifiers that can choose a class c from the set C, they define the disagreement on example e by the *vote entropy* of the pool:

$$D(e) = \sum_{c \in C} \frac{V(c,e)}{k} \log \frac{V(c,e)}{k}$$

where $V(c,e)$ is the number of members of the committee that assign the class c to the example e. When sequential selection is performed on data from an infinite stream, like in the QBC algorithm, Argamon and Dagan propose to either query all the examples for which $D(e)$ is about some threshold or to randomly select an example for labeling based on the flip of a coin biased according to the vote entropy –a higher vote entropy leading to a higher probability of selection. The two methods are called *thresholded* and *randomized selection*, respectively. For pool-based active learning is performed, the authors simply select the example with the highest vote entropy value.

A limitation of the method proposed in **[2]** is that it ignores the additional information provided by classifiers with probabilistic outputs. In a binary $(+,-)$ classification problem where the output of the classifiers represents the probability of the $+$ label, the vote entropy on an example on which a three member committee produces outputs $(0.51, 0.51, 0.49)$ is the same as if the outputs were $(0.99, 0.98, 0.01)$. The vote entropy score ignores the classification confidence information.

A solution to this limitation was introduced by McCallum and Nigam in **[49]**, where they suggested to use the *KL-divergence-to-the-mean* proposed in **[65]** as a disagreement method. The KL-divergence-to-the-mean is defined as the average of the KL-divergences between the posterior class distribution of each committee member and the average posterior class distribution of the entire committee[4]. If k is the number of members in the QBBAG committee, x is an unlabeled example, the KL-divergence-to-the-mean is defined as

$$\frac{1}{k}\sum_{m=1}^{k} D(\mathrm{P}_m(C|x)||\mathrm{P}_{avg}(C|x)),$$

where C is a random variable over the classes, $\mathrm{P}_m(C|x)$ is the posterior class distribution of committee member m, and $\mathrm{P}_{avg}(C|x)$ is the average posterior class distribution of the committee. The KL divergence between two distributions $\mathrm{P}_1(C)$ and $\mathrm{P}_2(C)$ is given by

$$D(\mathrm{P}_1(C)||\mathrm{P}_2(C)) = \sum_{j=1}^{|\mathcal{C}|} \mathrm{P}_1(c_j)\log\frac{\mathrm{P}_1(c_j)}{\mathrm{P}_2(c_j)}$$

where $\mathcal{C}$ is the set of classes, and $\mathrm{P}(c_j)$ is the probability the class label is j.

[4]We will follow here the development from McCallum and Nigam **[49]**

KL-divergence-to-the-mean correctly takes into account the confidences produced by classifiers with probabilistic outputs, and as a result is able to distinguish between examples on which the classifier committee would output $(0.51, 0.51, 0.49)$ and $(0.99, 0.98, 0.01)$.

Considering that the KL-divergence-to-the-mean is a more principled measure of committee disagreement, we chosen it to replace the original method proposed by Abe and Mamitsuka [**1**] for all our experiments involving the Query-by-Bagging algorithm. We have also directly compared the performance of the algorithm using vote entropy and the KL-divergence-to-the-mean, confirming that the more principled scoring method leads to better results.

5. Uncertainty Sampling

One of the best known active learning algorithms –which is also a heuristic alternative to QBC– is *Uncertainty Sampling* (US), a very simple algorithm proposed by Lewis and Gale[**45**] in 1994. The uncertainty sampling algorithm requires using a learner that can produce hopefully reasonable estimates of its prediction confidence. The classifier is trained on all the labeled data available, and then is applied to all the unlabeled examples. The unlabeled data point that gets classified with highest uncertainty (hence the name) is selected for querying (see Figure 2.8). As we have indicated earlier, the Query-by-Boosting algorithm can be viewed as being very similar to Uncertainty Sampling.

The motivation for uncertainty sampling comes from the desire to avoid having to sample classifiers from the version space. Lewis and Gale propose to approximate the *classifier uncertainty* (the confidence that one of the labels occurs given the current version space) by the *label uncertainty* as estimated by the unique classifier trained on all labeled data. As we show in Figure 2.9, this approach can work very poorly when the classifier used underestimates its uncertainty. Consider a classifier that randomly selects a hypothesis from the version space, and based on that single hypothesis falsely indicates that it has a confidence of 90% in its classification of the pink point (farther from the decision boundary than the blue one). In reality, given

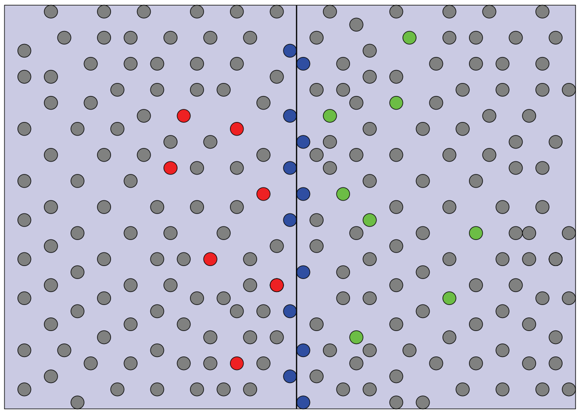


FIGURE 2.8. The Uncertainty Sampling algorithm fits a classifier to the currently available labeled data (the red and green points), and selects for querying the unlabeled data points that have the highest classification uncertainty, i.e. are closest to the decision boundary (the blue points).

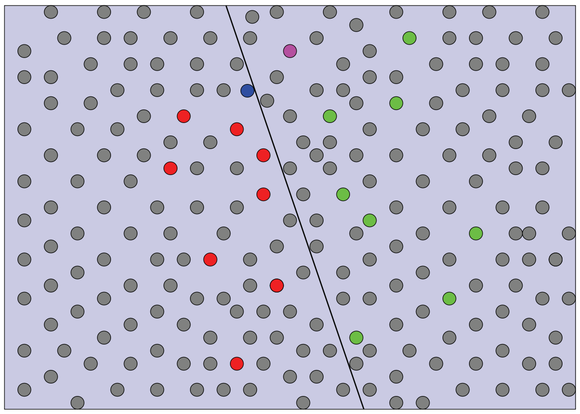


FIGURE 2.9. The Uncertainty Sampling algorithm can work poorly if the learning algorithm underestimates its uncertainty, because it ignores the uncertainty in the classification boundary and only looks at the uncertainty in the label instead. Query-by-Committee would not be affected by this problem.

the labeled data available, the uncertainty in the label of the pink point should be larger than the one of the blue point. If a QBC committee analyzes the same problem with the same base learner, it is likely that more of the hypotheses it samples from the version space disagree on the label of the pink example than on the blue one, and as a result a more informative query will be made. This problem is also acknowledged by Lewis and Gale [**45**].

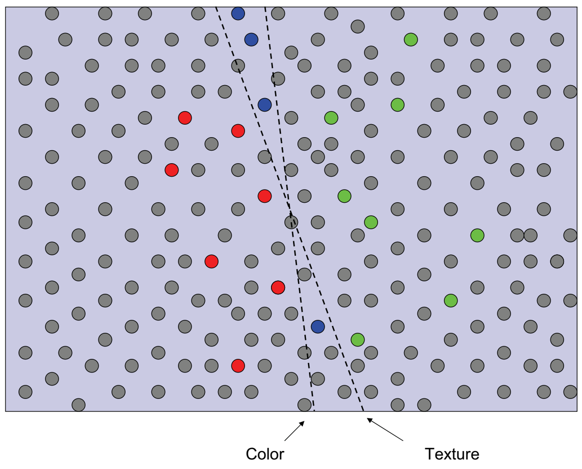


FIGURE 2.10. Co-Testing identifies unlabeled examples that are likely to lead to great classification improvement as those that get classified differently by learners trained on different redundant views of the data. The algorithm could either randomly select an example on which there is disagreement, or it could select the example on which the views disagree with the highest confidence.

Since uncertainty sampling is such a well-known algorithm and is used in many active learning publications as a baseline, we have decided to include it in our experiments.

6. Co-Testing

A more interesting algorithm is Co-Testing (Muslea et al. [**59**]), an algorithm that borrows the idea of using *redundant views* from Co-Training (Blum and Mitchell [**9**]) and adapts it to the active learning domain. According to the authors, "a domain has redundant views if there are at least two mutually exclusive sets of features that can be used to learn the target concept". Since in our domain of interest one often uses different sensing modalities for perception, we have considered the features extracted from the different sensors to be our views. Thus, in our experiments we will classify obstacles based on "color", "texture" or "laser" views, for example.

The algorithm works by training the different views on the available training data, and classifying the unlabeled data in all the views. The set of examples on which the views disagree represents the pool of potential labeling candidates (see

Figure 2.10), and different flavors of Co-Testing exist for selecting one query out of this pool. The authors claim that the most efficient gains can be obtained by querying contention points on which the views disagree most strongly, but they present experimental results in which they simply choose one contention point randomly. Comparing Co-Testing to Uncertainty Sampling, Muslea et al.[**59**] show that by using multiple views their algorithm can be more aggressive than US by querying contention points on which the algorithms are most confident, which increases the probability of a large effect in at least one of the views. For comparison to QBC, the authors construct a learning problem which is PAC learnable but on which QBC fails with high probability while Co-Testing succeeds in very few steps. In the experimental results presented without confidence bars, Co-Testing seems to generally perform quite similarly to QBBAG and QBBOOST.

We have chosen to include Co-Testing in the set of algorithms we analyzed, mostly due to the fact that in the obstacle detection domain it is often the case that multiple views of the data (even if not redundant) are available.

7. Other Active Learning Algorithms

As it is often the case when a relatively large research area needs to be condensed to a few algorithms while also taking into account domain constraints, there are a number of very interesting active learning algorithms we have left out of our discussion and experiments. Most of the algorithms we presented can be related quite directly to the QBC algorithm and its sampling approach to determining the splits induced in the version space by each unlabeled example, so we will now discuss two slightly different techniques.

In 2000, Tong and Koller[**86**] have proposed an active learning algorithm which attempts to more directly estimate the reduction in the size of version space by exploiting properties of support vector machines (SVMs, [**88**]). They start by assuming that the training data is linearly separable in the induced feature space produced by the kernel employed by the SVM; this assumption is reasonable, given that most kernels project the data in a higher dimensional space. The authors then show that the hyperplane produced by the regular SVM training algorithm corresponds to a parameter vector at the center of the largest radius hypersphere that can be fully embedded in the version space. They also show that each training example represents a hyperplane in the parameters space, and that the version space is given by the intersection of all the hyperplanes generated by the training data. The training examples whose corresponding hyperplanes touch the hypersphere are support vectors, and the radius of the hypersphere is the margin of the SVM.

Based on this observation, Tong and Koller propose three variations of learning algorithms. The first, called “Simple Margin”, tries to identify the unlabeled example which corresponds to the hyperplane that is closest to the center of the maximum radius hypersphere. A second version, called “MinMax Margin”, is trying to alleviate the assumption made by the “Simple Margin” algorithm, which is that the version space is relatively symmetric. Since each hyperplane (i.e. unlabeled example) that splits the version space in two, “MinMax Margin” tries to select an unlabeled example such that after the maximum radius hypersphere is fit in each section, the minimum of the two radii is maximized. A slight variation of “MinMax Margin” is “MaxRatio Margin”, which instead looks at the ratio of the two hypersphere radii.

Although Tong and Koller present relatively good results compared to random sub-sampling in the text classification domain, we chose not to experiment with their methods because they depend on the use of SVMs, which are still impractical to train on data sets of the size that we are interested in. Nevertheless, we consider the algorithms proposed in [**86**] well principled and an important contribution to the field.

The last active learning method we would like to discuss is the one proposed by Roy and McCallum in [**76**]. The authors point out that most of the existing active learning algorithms attempt to reduce the size of the version space, but they do not consider the *true* evaluation criterion, which is the reduction in the error rate on future test examples. While most other researches discarded this criterion from the start because of the large computational requirements expected [**27**], Roy and

McCallum demonstrate that the approach can be made practical through the use of Monte-Carlo sampling and learning algorithms that have very efficient incremental training. Presented very succinctly, their algorithm consists of the following steps:

1. consider all unlabeled examples x in the pool as query candidates
2. for each unlabeled example, consider all its possible labels y
3. temporarily add the (x, y) pair to the labeled data set, and train a classifier on it
4. using the probability distributions induced by the classifier over the labels of all the unlabeled data, estimate the expected future loss[5] if the example x is labeled and it receives label y.
5. the losses expected for the different labels y of the current example x are accumulated using the distribution over y produced by a classifier trained on the "truly" labeled data set (without the example x and its guessed labels. This will give the loss expected if the example x is selected for labeling.
6. select for labeling the example for which the decrease in the expected loss is the largest.

As presented here, the algorithm would be impractical due to the amount of computation required, but two different rounds of sampling can help control the computational complexity. First, the authors argue that the pool of candidate queries (the unlabeled data set) can be reduced by random sub-sampling, or some other form of pre-filtering. Secondly, when estimating the expected loss after labeling a specific instance, one could again subsample the pool. The authors also show that although frequent retraining of the classifier is required, only a single training example is added at each time, which means that their active learning algorithm could benefit greatly from using learners with efficient incremental re-training procedures, such as Naïve Bayes or SVMs. Similarly to the authors of [**86**], Roy and McCallum present good results on typical text classification data sets of approximately 1000 documents.

[5]The authors present derivations for both log-loss and the 0/1 loss

We consider this algorithm to be one of the most elegant and well founded active learning algorithms, but for the reasons we presented in Figure 2.6 (page 18), we believe it would perform very poorly in our domain: the outdoor perception problem often results in extremely unbalanced class priors, which would make both sub-sampling steps required for efficiency very unlikely to result in meaningful estimates. Furthermore, we would like our active learning algorithms to relatively easily extend to *very* large data sets of hundreds of thousands of examples, which is not the case of the one described in [**76**].

CHAPTER 3

Active Learning: Challenges in Outdoor Perception

1. Unlabeled Data Filtering

We have presented a number of active learning algorithms which all try to reduce the uncertainty in the location of the decision boundary. There is one aspect, common to all the approaches we described, which has been completely ignored so far: their initialization.

Any typical active learning algorithm needs to be initialized with some small amount of labeled data, which is required before being able to reason about decision boundaries, version spaces and expected losses, as we have shown in the previous section. The initialization data is a very important factor for the degree

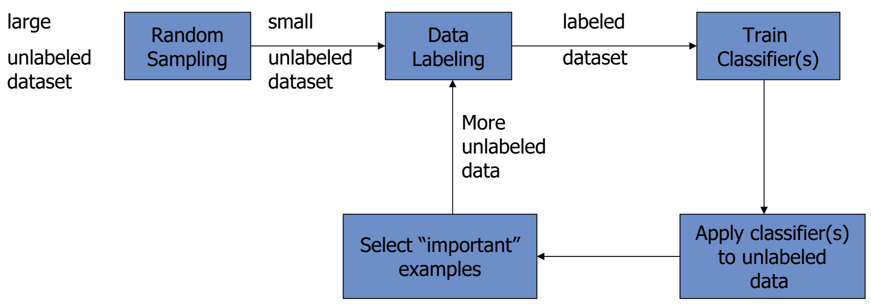


FIGURE 3.1. A pictorial representation of a typical active learning algorithm. In this diagram, the initialization problem is solved by performing random sampling.

of success that can be expected from the future active learning rounds. If, for example, the initial estimate of the version space is completely wrong, it is irrelevant what technique is used for reducing its size: the active learning algorithm will still perform poorly.

As a result, the question of how the initialization data is selected is very important. For the majority of the active learning algorithms proposed in the literature, the solution adopted is the one represented in Figure 3.1, which is to randomly sample a few unlabeled examples from the large pool available, label and then use them for initialization. In a few other cases, a human expert selects manually the examples that are to be used for initialization.

For the problem domain in which we are interested to use active learning, neither of the two solutions is practical. Randomly selecting data for initialization will fail for the same reasons for which random sub-sampling is not a general solution for data set reduction[1]: for very unbalanced data sets, it is likely to obtain a sample whose distribution is dramatically different from the one of the initial, large data set. For example, for an "obstacle"/"non-obstacle" classification problem that is highly unbalanced, random sub-sampling could generate an initialization set that only contains "non-obstacle" examples. In such cases, the performance of any active learning algorithm is necessarily bad. This situation occurred frequently in our experiments.

Manually selecting data for initialization avoids this type of problems, but it does not scale well. For very large data sets, it can be very inefficient to have a human expert analyze the data and guess which examples would lead to good active learning performance. Furthermore, one of the key goals of our research is to make the use of learning based systems possible for users with very little training. Being able to reason about the impact of the data used for initialization on the future performance of the system is certainly outside of the scope of tasks that can be performed with very little training.

[1]see Section 1 on page 13

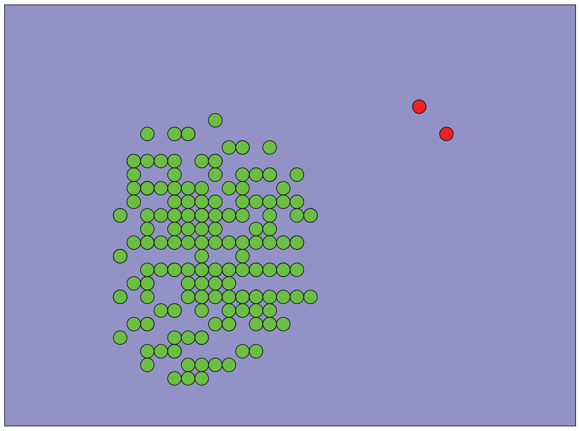


FIGURE 3.2. The majority of the classification problems encountered in mobile robotics applications consist of unbalanced data sets where the most frequent class (for example, "non-obstacles") has a representation in the feature space that is quite compact and thus densely populated. The examples from the infrequent class tend to appear in very sparsely populated regions of the feature space.

The solution we propose for the initialization problem is the Unlabeled Data Filtering algorithm (UDF **[18]**). The observation that led to the development and testing of UDF was the fact that in most classification problems of interest in the mobile robotics domain, we are confronted with unbalanced data sets which have the additional property that the most frequent class is usually quite compact.

This situation, depicted in Figure 3.2, occurs because a high percentage of any data set recorded with a mobile robot contains examples of the terrain the vehicle can safely drive over: roads, fields of grass, small forest vegetation. Given the relatively uniform appearance of these types of terrain compared to the highly variant signature that obstacles could have, we hypothesized that most of the content of our large, unlabeled data sets was *redundant*. In order to learn the appearance of a field of grass, for example, one only needs to see a few examples that cover well the region of the feature space corresponding to the "grass" class. There is no benefit to be gained from obtaining labels for massive amounts of additional "grass" data that projects to the same region of the feature space, which is already populated. On the other hand, when we observe an unlabeled example whose signature in the feature space is significantly different from the rest of the data, we can hypothesize that it corresponds either to some type of obstacle or to some new type

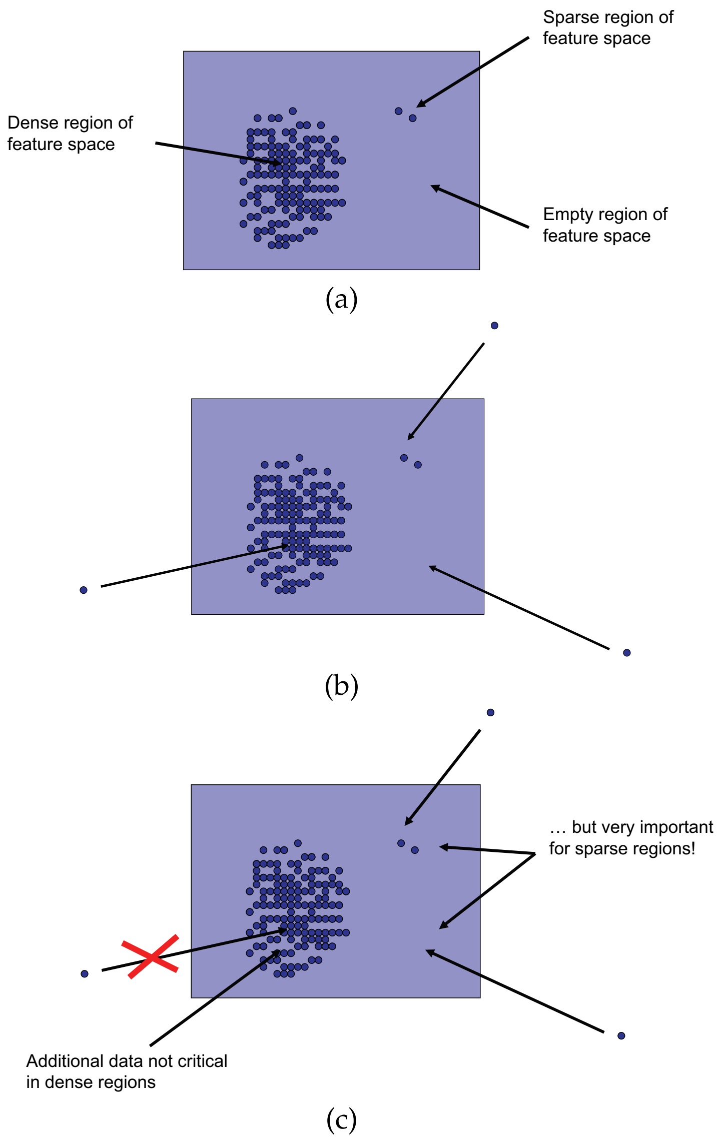

Sparse region of feature space
Dense region of feature space
Empty region of feature space
(a)
(b)
… but very important for sparse regions!
Additional data not critical in dense regions
(c)

FIGURE 3.3. The Unlabeled Data Filtering algorithm. (a) At time T, some amount of data (all the points in the figure) is already selected for labeling. (b) At time $T + 1$, new data is observed. Each data point is projected into the feature space. (c) The data points projecting into regions of the feature space that are already densely populated are ignored, while the rest are added to the set of data points selected for future labeling.

of “grass” that has not yet been observed, and we would like to query for its label. This simple intuition, represented graphically in Figure 3.3, is all that is required for understanding the UDF algorithm.

As its name suggests, we consider UDF more of a filtering method than an active learning algorithm, although we have obtained interesting results when using

```
1. One or more data points are randomly selected for initialization, and
   added to the initially empty data set of points selected for labeling, $S$.
2. For as many iterations as data points we want to select for labeling
   a For each unlabeled example $x$ from the unlabeled data pool $U$, estimate
     $p_S(x)$, the density function induced by the set of selected points $S$ over
     the feature space at location $x$
   b Select for labeling the unlabeled data point $x_{min}$, where $x_{min} = \min_{\forall x \in U} p_S(x)$
   c Set $S = S \cup x_{min}$ and set $U = U \setminus x_{min}$
```

FIGURE 3.4. Pseudo-code for the Unlabeled Data Filtering algorithm. Notice that unlike typical active learning algorithms, UDF is quite insensitive to the initialization data set: *some* data is only required so that the first estimation of the density function $p_S(x)$ is valid.

it as a stand-alone active learning technique. UDF's filtering scheme is trying to uniformly spread data over the entire feature space, by selecting data points in the least densely populated regions of the feature space. Equivalently, we could say UDF filters the data points that provide information that is likely to be redundant with what is already in the selected data set. It is important to notice that although UDF selects data points for labeling, the labels of the selected examples are not used *at all* by the algorithm: only the location in the feature space of the data points is important. Unlike a typical active learning algorithm which continuously adapts its behavior based on the result of the queries it makes, UDF can be used as a completely non-interactive method that selects an entire batch of data for labeling at a later time. The steps of the algorithm are enumerated in Figure 3.4.

As one would expect, the challenging part of our algorithm consists in reliably and efficiently estimating the data density in various regions of the feature space. Several standard techniques are available for this purpose, and the ones we considered are kernel density estimation (KDE) and mixture models, such as mixtures of Gaussians (MoGs)[**7, 20**].

Kernel Density Estimation is a non-parametric probability density modeling technique, also known as *Parzen window* estimation. Assuming that a random sample $x_1, \ldots, x_N$ is drawn from a probability density $f_X(x)$ and that we are interested in the value of the probability density function at location x, the estimate provided by KDE is

$$\hat{f}_X(x_0) = \frac{1}{N\lambda} \sum_{i=1}^{N} K_\lambda(x_0, x_i) \tag{3.1}$$

where $K_\lambda(x_0, x_i)$ is kernel function whose value decreases as the distance between x_0 and x_i increases. The parameter λ, the bandwidth of the kernel, controls how fast the value of the kernel function decreases. In essence, the KDE estimate is a weighted count of the data points in a neighborhood of x_0, where the points closer to x_0 receive more weight. There are many possible options for the kernel function, but the kernel choice is in general less important than choosing the right bandwidth, which controls the trade-off between bias and variance of the KDE estimate. In our experiments, we use the Gaussian kernel, and we search for the optimal bandwidth setting using cross-validation.

Another way to look at kernel density estimation is to consider that a (Gaussian) kernel is placed over each one of the points x_i in our data set, and their contributions at location x_0 are averaged. This interpretation of KDE makes it easy to connect it to the estimation methods based on mixture models: in the case of a mixture of Gaussians, we attempt to model the density function $f_X(x)$ using less than one Gaussian per data point. A mixture of Gaussians model with M components estimates the density at a point x_0 as

$$\hat{f}_X(x) = \sum_{m=1}^{M} w_m \phi(x; \mu_m, \Sigma_m) \tag{3.2}$$

where w_m,μ_m and Σ_m are the weight, mean and covariance matrix of the m^{th} Gaussian. Unlike kernel density estimation, mixture models require a fitting step in which the parameters of each Gaussian in the mixture model are estimated, and their weights are computed. The algorithm of choice for fitting mixture models is Expectation-Maximization [**17**].

Although both kernel density estimation and mixture models can be used with the UDF algorithm, their properties makes them fit for slightly different types of data sets. Compared to mixture models, kernel density estimation has the advantage that no fitting process is required, and it is not necessary to select a number of mixture components to use. The only tunable parameter is the bandwidth, which can generally be chosen without too much difficulty using cross-validation. For mixture models, one has to choose the number of components, and then fit the model using an algorithms which only find a local minimum. Determining the "optimal" number of mixture components through cross-validation is certainly possible, but given the risk of having EM converge to a bad local minimum, it is usually necessary to perform a large number of fits which can be time consuming.

Despite this challenging fitting process, mixture models require far less computation for each estimate of $\hat{f}_X(x)$ than kernel density estimation: $O(M)$, where M is the number of mixture components, instead of $O(N)$, where N is the number of data points. Especially for large data sets, the difference becomes very important.

The choice between KDE and MoGs becomes a choice between having a cheap fitting process but computationally expensive density estimates (KDE), or an expensive fitting process and cheap density estimates (MoGs). Since the fitting process is only required when new data is added to the set of data points selected for labeling, the factors that influence the KDE/MoG choice are the following:

- the larger the number of data points that we want to select for labeling, the more attractive KDE is.
- the larger the pool of unlabeled data to be considered, the more attractive MoG is.

We have performed UDF experiments with both KDE and MoGs, and for the data sets we considered we preferred to use kernel density estimation. The performance of the UDF algorithm does not heavily depend the density modeling choice. The speed of the algorithms was sufficient for our purposes, and several types of specialized data structures can be employed for achieving significant speed-up factors [**31, 54**].

There is an additional challenge in estimating density functions, which affects both methods: the high dimensionality of the input space. Although it is not intuitively obvious, it can be demonstrated [**7**] that as the dimensionality of the input space increases, almost all the points in a data set will be very far from each other. This *curse of dimensionality* makes both KDE and MoGs unusable in high dimensions.[2] In our problem domain, the feature vectors used for classification are often high dimensional, since they typically include features extracted from several sensing modalities. The solution we employed for addressing the dimensionality issues was the standard procedure of compressing the data using principal component analysis (PCA). We will discuss in more detail the effects of the dimensionality reduction step in Section 4.3 of Chapter 4.

The Unlabeled Data Filtering algorithm is the connection between our work and data mining. The idea of using some form of probability density estimation in order to identify sparsely populated regions of some feature space is certainly not new: it is in fact a standard technique in the anomaly detection literature. Points that are located in regions where the density of data points is small can be interesting for many reasons, depending on the space in which they are defined and on the distance metric used. If each data point is a normalized string of tokens representing the UNIX commands typed by a user, one could use the minimum string distance from some set of sample "valid" sessions as an indicator for illegal use of a computer account (see for example [**44**]). If the data points are vectors describing autoregressive models fit to different portions of a time-series, the distance between a vector describing the most recent data and the vectors corresponding to already observed patterns can be used for discovering anomalous regimes in multivariate time-series [**5**]. Many other applications exist, as varied as data cleaning or detecting employers with poor injury histories.

[2]The number of dimensions where difficulties start occurring depends on the size of the data set, but is quite low: five or six dimensions are already very challenging.

The approach that is closest in spirit to ours[3] was described by Pelleg and Moore in [**64**]. Their application of interest is the retrieval of interesting data patterns from huge data sets of telescope images. Other than their overwhelming size, the data sets of interest are characterized by a small number of anomalies (0.1% of the data), representing data that is not well described by existing theories and models. Of these anomalies, 99% correspond to imaging artifacts (referred to as "boring" anomalies), while 1% (0.001% of the original data set) represent "useful" anomalies, corresponding to objects worthy of further research. The authors are interested in identifying the 0.001% of the data that should be presented to astrophysicists for further analysis. They start from the assumption that it is easy to separate all the anomalies (the boring and useful ones) from the rest of the data, since they do not fit well the existing models. The problem of interest is to sort the anomalies in such a way that the "useful" anomalies are likely to be presented earlier to the user. To do this, they apply the typical pool-based active learning framework in batch mode.

After careful consideration of the paper, it is our understanding that the input data available for labeling is *all* the data set, not only the anomalies. The authors argue that the human expert can easily label data with a lot more detail than just "interesting/non-interesting". Instead, the expert can categorize the regular data, and identify the specific cause of the anomalies. Of course, this approach can be always scaled down to a "interesting/non-interesting" problem. From this point on the authors no longer refer to the problem of discriminating between interesting and boring anomalies: they only have a large active learning problem, and hope that examples from one of the classes –the interesting anomalies– will be selected as queries sooner rather than later.

What makes Pelleg and Moore's approach very similar to ours is that they are also confronted with an active learning problem where the classes are extremely unbalanced. Similarly to our very own experience, they state that

> "in data sets with the rare categories property, this [using random sampling for initializing active learning, n.n.] no longer

[3]But developed completely independently, we believe

holds, and much of our effort is an attempt to remedy this situation".

Our approaches for addressing this problem are quite different, however. First, Pelleg and Moore start from the *entire* unlabeled data set and use *semi-supervised learning* to model the joint labeled and unlabeled portions of it at each iteration using a mixture of Gaussians. In UDF, when computing the kernel density estimate at new data points, we only use the data that was selected for labeling until the current iteration. This, we believe, is an important difference we will return to later.

After fitting a mixture of Gaussians to *all* the data, the authors ask a human expert to label the 35 strangest examples, where "strange" is defined based on a few different metrics:

- Choosing points with low likelihood, as determined by the mixture of Gaussians
- Choosing ambiguous points, which are points disputed by the different classes
- Combining (alternating between) unlikely and ambiguous points
- Interleaving, a method based on "dissociating" the different components of the Gaussian mixture model.

One can easily recognize the first option as equivalent to what UDF does (except for the different manner of estimating the probability density function) and the second as a version of Uncertainty Sampling[4]. Using artificially generated data, the authors show that the first three methods can fail to query for points from an interesting cluster until almost all other data points are labeled. This is clearly undesirable for the type of real data they are interested in using.

The `interleave` algorithm they propose, admittedly without "further theoretical justification", is a novel approach consisting in departing from the way in which the mixture of Gaussians is typically used to estimate the likelihood of the

[4]In the paper, the authors also mention a similarity "in spirit" with the Query-by-Committee algorithm, which we believe is quite different

points. Instead of using all the mixture components with their respective weights to compute the likelihood at any data point, Pelleg and Moore propose to use only the component that "owns" the data point to identify anomalies. Then, to create a list of anomalies to present to the user, they iterate through all the components of the Gaussian mixture, and select from each component the data point that has the lowest likelihood, as computed based on that unique component. The paper presents results on both artificial and real data sets. On the five real data sets presented, the `interleave` algorithm performs:

- similarly to global low likelihood selection (1 time)
- slightly worse than global low likelihood selection (1 time)
- similarly to the alternation between ambiguous and low likelihood points (1 time)
- better than low likelihood selection (1 time)

Since only one run of the algorithms is presented on each data set without confidence intervals, it is essentially not possible to draw any strong conclusions about the different methods. Assuming that the authors' conclusion that the "local" low likelihood selection method (the `interleave` algorithm) is better than the global one is correct, our belief is that the explanation rests in the fact that global selection tends to select more extreme anomalies, while the "local" method has a chance to select anomalies that are not as extreme. Logically, one would expect anomalies generated by artifacts in the imaging process to be more extreme than anomalies corresponding to new stellar objects. Our hypothesis is very much reinforced by the authors' comment that they found it to be beneficial to add a uniform-density "background" component to the Gaussian mixture, that "nominates hints more often than any other component". We cannot help noticing that nominating hints from a uniform density over the feature space –essentially ignoring the high density clusters of data points– is *precisely* what our Unlabeled Data Filtering algorithm does.

We will conclude this protracted discussion of the relation between UDF and anomaly detection by simply restating the main characteristic of our algorithm:

although it uses the same “tool” as anomaly detection, which is density estimation, UDF does *not* look for anomalies, but instead tries to uniformly cover the feature space with data points. This difference comes from the fact that in UDF we only model the density of the points already selected for labeling (in our case), while anomaly detection models the density of the *entire* data set available.

2. The Data Block Constraint

We have presented several active learning algorithms, which, once initialized, should in theory be effective at reducing the amount of labeled data required to achieve good predictive performance. We have also proposed an algorithm for generating good initialization data sets. It might seem we are ready to evaluate the applicability of the active learning approach to the obstacle detection domain, but in reality one last challenge prevents us from doing so: the *data block constraint*. To introduce this concept, we need to present a few details about the structure of our obstacle detection system.

Although the purpose of obstacle detection is to identify the 3-D locations that need to be avoided by a vehicle, the classification in the “obstacle”/”non-obstacle” classes does not take place at the voxel level in our case. Instead, using range data and known 3-D transforms between the different sensors on the vehicle, all the data is projected in the frame of one of our imaging sensors, as described in Figure 3.5.

The motivation for the choice of performing the labeling and classification in 2-D, at the image level, is convenience: it is easier and faster for a human to analyze a color image and identify the regions that correspond to obstacles, than to look at point clouds, orient them correctly and then try to identify and mark the 3-D regions that should be avoided. Our impressive capability of quickly identifying obstacles in 2-D images comes from the fact that this is the precise task that the human visual system has been tuned to perform well over millions of years; our experience with 3-D point clouds is much shorter.

As we show in Figure 3.6, working in the image frame has the additional benefit that the labeling process is very simple: the human expert needs to only circle

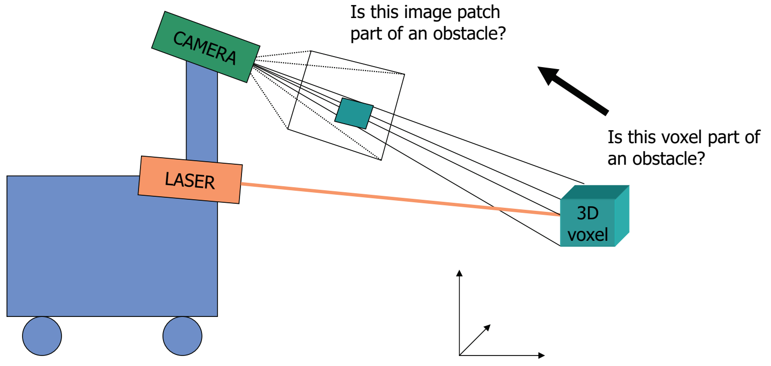


FIGURE 3.5. The problem of interest is to distinguish between 3-D locations that are traversable or not. In order to render the labeling process convenient, we project all the data from the 3-D world into the frame of one of our imaging sensors, transforming the task of classifying 3-D voxels into the task of classifying 2-D image patches. The range measurements about the scene and the known 3-D transforms between the frames of the different sensors make it possible move between the voxel and patch representations.

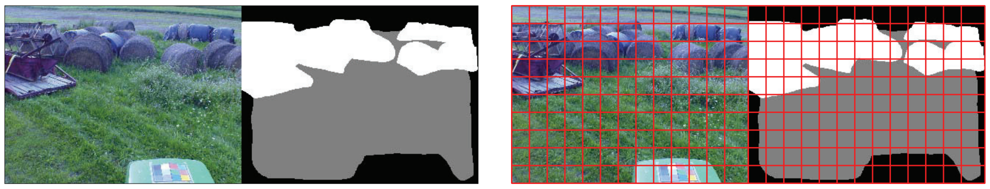


FIGURE 3.6. The labeling process in the image frame is easy: the human expert only needs to (approximately) trace the contour of the classes of interest that are represented in the image (left). Once a label image is generated, the color and label images are split into a grid of patches, and the feature vector x extracted from the sensor measurements associated with each patch in the color image gets associated with the label y extracted from the corresponding label patch.

regions that correspond to the different classes of interest[5]. Both the color and the label images get split into a grid of patches, and the feature vector x extracted from all the sensor data associated with each color patch can be associated with the target label t indicated by the corresponding label patch. Thus, from labeling a single image, a large number of training examples (x, t) can be extracted. At run time, the trained classifier is presented with new feature vectors x_i and produces the corresponding predictions y_i. Using the range information associated with each patch, these predictions can subsequently be mapped back to the 3-D world.

[5]for example with a tablet device

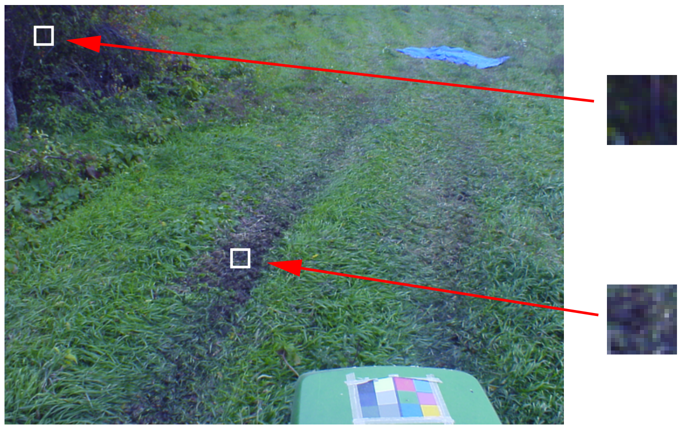


FIGURE 3.7. The contextual information contained in an image is crucial for the ability of a human expert to label patches. Just based on their visual appearance alone, it is difficult to label the tree patch as "obstacle" and the ground patch as "non-obstacle". As a result, every time labeling takes place, an entire image must be presented to the expert, with the benefit that a large number of image patches get labeled at once.

The fact that labeling takes place at the *image* level but the classification process at the *patch* level is a problem for active learning algorithms, because they operate at the same level as the classifier and thus propose *patch* queries instead of *image* queries for labeling. For the reasons illustrated in Figure 3.7, the human expert will still need to be presented with an image to label, despite that fact the active learning method only makes a patch proposal. As a result, some method for going from patch interest scores (an output of most active learning methods) to images is required.

The fact that there is some structure in the unlabeled data set (the images) which links certain data points (patches in our case) together, is what we refer to as the *data block constraint* on active learning: algorithms need to choose between fixed groups of data points, although they reason about individual examples. This situation is depicted in Figure 3.8.

Given that active learning produces an "interest" score for each unlabeled patch, a simple solution for the data block constraint would be to always label the image that contains the most "interesting" patch. The problem is that this approach can work poorly because often, the most unusual data patterns observed are outliers caused by various data artifacts. If one is only using color information,

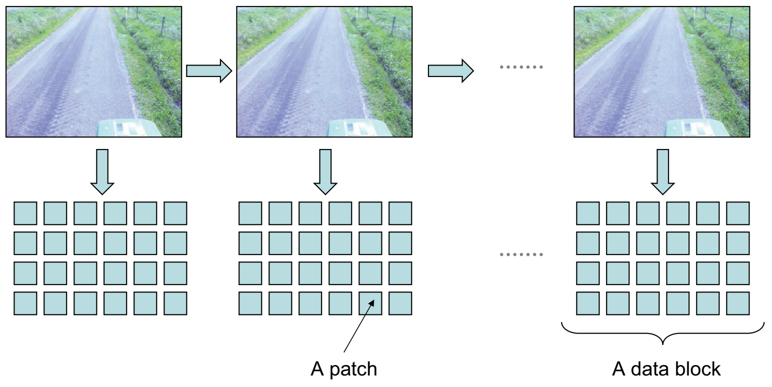


FIGURE 3.8. The Data Block Constraint: the unlabeled data has a fixed structure that is above the level considered by the active learning algorithms. Active learning typically reasons about individual or unstructured batches of points, whereas in this case there are several *fixed* blocks of data from which one needs to be selected for labeling. Generally, all the data points in the selected block will receive labels in one labeling step.

for example, we would not want to label an image just because a spot of specular reflection appears somewhere in the image. Furthermore, only basing the interest score of an image on its most interesting patch would prevent us from distinguishing between a scene containing a single obstacle and one with several.

As we have argued in [**18**], the opposite alternative, which is to average the interest scores over the entire image, is also undesirable: high-interest patches corresponding to small obstacles could get overwhelmed by large amounts of uninteresting patches. A good example to think of is an image of a rock on a golf course: most patches in a picture correspond to the relatively uniform (and thus not very interesting) grass, while a few come from the obstacle.

To balance between these undesired options, we propose two solutions: to aggregate the patch interest measures over some small and *fixed* amount of patches (usually 1-3% of the available patches in any image), or to dynamically determine the number and identity of patches determining the score.

The first solution addresses the outlier problem by accumulating the interest measure over several of the highest scoring image patches. In Figure 3.9 we have a stream of image data blocks, and, at the bottom, a representation of a common image patch queue containing *all* the image patches in the unlabeled data set, sorted

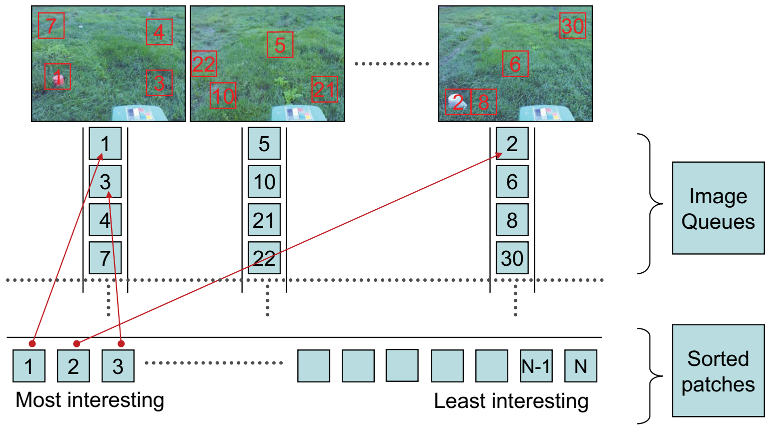


FIGURE 3.9. Aggregating the interest score of each image over a fixed number of patches. All the image patches in the unlabeled data set are labeled and sorted in a common queue represented at the bottom of the figure. The patches from the common queue are then assigned, in order, to the images they originate from. The process is repeated until the individual queue of each image contains at least K patches, and their scores can be averaged. The image with the highest average gets selected for labeling. Note that instead of sorting the entire set of unlabeled data patches, one can equivalently sort the scores of the patches extracted from each image.

by the interest score assigned to them by the active learning algorithm[6]. Instead of simply selecting for labeling the image containing the patch at the top of the queue, we define instead an empty patch queue for each of the images in our data set, and proceed with the following algorithm:

1. Pop the image patch that is at in front of the common sorted queue, and place it in the queue of the image that it originated from.
2. Repeat the process until all the image data blocks contain at least K patches in their queue.
3. Compute the accumulated interest score of each image in the unlabeled set by averaging over the K interest patches in its corresponding queue.
4. Select for labeling the image with the highest accumulated interest score.

Notice that although the accumulation algorithm we presented requires a sorting step involving all the data points in the unlabeled set, in reality this expensive step can be avoided: since after sorting the patches are distributed in the order of

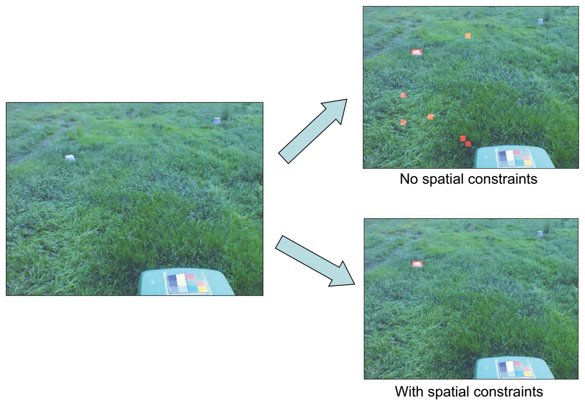
[6]Let us assume for now that sorting the few million patches in our data sets is not a problem.

the score to their corresponding source images, we could achieve the exact same configuration of the image queues by performing many small local sorts that only involve the data from one image. As a result, this simple algorithm scales well to very large data sets. The only parameter that needs to be manually chosen is K, the number of patches over which we aggregate the score. Our experiments have shown that identifying a good setting of K is not difficult: a relatively wide range of values result in equally good performance.

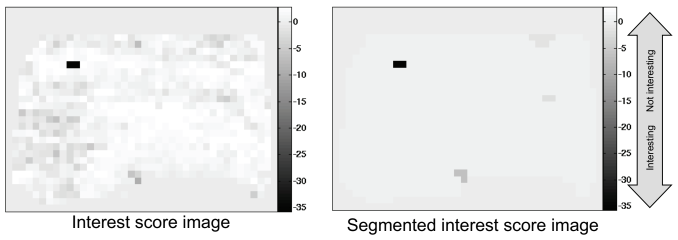
Although the accumulation scheme we have just described works very well in practice, the fact that we are aggregating interest scores over a *fixed* number of image patches without any spatial constraints imposed can be bothersome: we could very well compose the scores of patches with high interest scores corresponding to an obstacle with patches of non-interesting grass that are located at various locations across the image (see Figure 3.10). In this case, we would suffer on a smaller scale from the same problem that makes it undesirable to accumulate scores over the entire image: the high scores of the "obstacle" patches can be be overwhelmed by data with lower interest scores.

The solution we propose for this problem is to take into account spatial constraints between the image patches whose scores get combined: intuitively, patches that are not near each other do not correspond to the same object and should not be combined. In Figure 3.10(a) we present an image of a scene containing a very small obstacle on a field of grass. In the upper-right image we show in red the image patches whose scores get combined when we force the algorithm to use eight patches to compute the image score. As predicted, the method selects the two interesting patches corresponding to the small obstacle, but then continues to select six more grass patches with much lower scores, that are spread over the image. The result we would prefer is to only accumulate the scores of the patches corresponding to the obstacle, as shown in the lower-right image in Figure 3.10(a).

We choose to impose spatial constraints by segmenting the *score image* represented by a 2-D lattice of the same dimensions as the patch grid, in which the value at each site is given by the interest score assigned by the active learning algorithm



(a)



(b)

FIGURE 3.10. Aggregating the interest score over a region of variable size. (a) The image on the left contains a small obstacle (upper left corner) in a grass field. When the aggregation scheme we described previously is forced to select eight patches for averaging, the method selects the two patches with high interest scores corresponding to the obstacle, but also six low-scoring grass patches in a spatially discontinuous configuration (upper-right image). Ideally, we would only like to take the patches corresponding to the obstacle into account (lower-right image). (b) The image score corresponding to the color image in (a). Darker values indicate patches that received a higher interest score. Using the Mean-shift algorithm, we segment the score image and select for aggregation the segment with the highest average interest score. In this case, this corresponds to the obstacle region.

to the corresponding patch (see Figure 3.10(b)). The segmentation step will insure that groups of image patches that (a) receive similar interest scores and (b) are geometrically contiguous get clustered together. Once the segmentation step is completed, the scores of the patches belonging to each image cluster can be averaged, and the image score can be chosen as the highest scoring average.

In our experiments, we chose to segment the 2-D lattices using Mean Shift[**12**], which is a well known and relatively robust algorithm used in the computer vision field for edge-preserving smoothing and segmentation. Mean-shift requires the manual setting of three more parameters. In our experimental section, we will discuss their sensitivity, and the importance of imposing spatial constraints in general.

To conclude this discussion, we would like to point out that the data block constraint does not only appear because of our decision to perform the labeling and classification steps in the frame of one of our imaging sensors. Even if one chooses to perform all the steps directly in the 3-D voxel domain, a human expert will still have to be presented with data from more than a single voxel in order to be able to correctly identify its class. It is also the case than once presented with several voxels of 3-D data, it is easy to label several of them at the same time. This is exactly the data block constraint, only that it is now applied to data points that are voxels instead of patches.

CHAPTER 4

Experimental Results

IN this section, we will present the outcome of a comprehensive analysis of the performance and applicability of the active learning techniques we described to the problem of outdoor perception for mobile robots. The section is structured as follows:

Subsection	Content	Page
1	A description and motivation for the set of questions we intend to answer based on our experimental evidence	57
2	The data modalities we use and the feature vectors we extract from them, along with a brief description of the main characteristics of each data log used	58
3	The performance metrics we employ to compare the different active learning algorithms	66
4	The experimental procedure: baseline methods, number of randomized trials, methods for introducing randomization	73
5	The results and their interpretation	81

1. Questions to Answer

As we have already indicated in the introduction, our research at the intersection of the active learning and mobile robotics communities is motivated primarily by a pressing need to render several supervised learning techniques practical for robotics applications. As a result, the most important question to investigate is whether the active learning "promise" holds *at all* in this domain. To the best of our knowledge, this effort is the first that investigates this important question.

To determine if active learning techniques are effective in our domain, we need to compare them to random and human performance. Is there anything to be gained by reasoning about the distribution of the pool of unlabeled data, compared to simply selecting data randomly for labeling? Even if this is the case, is the efficiency of any active learning algorithm comparable to that of a human expert, if we assume the human can analyze all the unlabeled data and manually select data for labeling?

An interesting question is also whether the active learning algorithms that seem more principled (such as QBBAG) perform best in our domain. We have introduced the Unlabeled Data Filtering algorithm as a solution to the initialization problem affecting most active learning algorithms. How effective is UDF at solving this problem ? Can it also be used as an unusual active learning algorithm ? If so, how does its performance compare to the one of the standard approaches ?

We have argued that the data block constraint sets our problem domain outside the realm of mainstream active learning applications. It is interesting to determine how successful the two solutions we proposed are, and how their efficiencies compare.

Last but not least, we need to determine if the methods we propose have good scaling properties: for our solutions to be practical, they need to be applicable to truly large data sets.

These are the types of questions we intend to answer in this section. At a high level, we hope to provide enough information to allow colleagues from our domain to estimate the applicability and robustness of the methods we presented.

2. The Data

The complexity and variation of the real world make it close to impossible to produce synthetic data that is of any use for outdoor perception problems. We have focused from the beginning on working with real-world data logs, the majority of which we collected using a CMU developed autonomous tractor. Although we

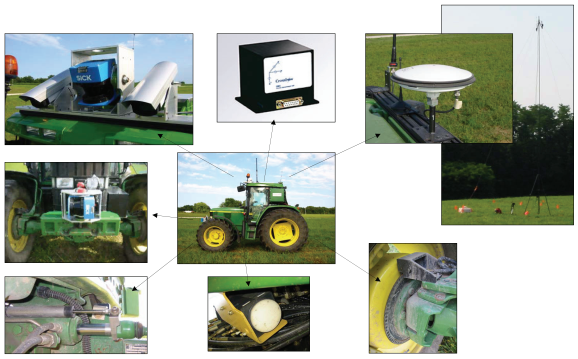


FIGURE 4.1. The autonomous tractor (center) and its sensors: a pair of high-resolution digital color cameras, two mechanically scanned SICK laser range finders, a steering potentiometer, a Doppler radar, four wheel encoders, a DGPS unit and three-axis optical gyro inertial measurement unit.

have performed active learning experiments with data collected with other platforms [**18**], we will only present here results obtained with data from the tractor. To insure that our observations are not likely to be related to the use of a particular vehicle platform, we perform most of the experiments on data extracted from three different subsets of the sensing modalities available.

The vehicle is equipped with the sensors presented in Figure 4.1 and an additional thermal infrared camera. The 3-D information from the laser range finders was projected into the frame of one of the color cameras, such that for each patch in the color image grid we had a list of 3-D points that projected into it. In the reverse direction, 3-D models enhanced with color information can easily be generated (see Figure 4.2). Having a precise position estimate makes it possible to accumulate 3-D measurements over several seconds, and recompute their projections into the image grid each time a new color image is available.

The 1280x960 color images are split into a 40x30 grid, resulting in patches of 32x32 pixels. The features we extract for each patch (see Figure 4.3) are the following:

Color: We represent each image patch in the perceptually uniform CIE LUV color space[1]. For each one the three color channels we compute the mean

[1]"Perceptually uniform" means that two colors that are equally distant (in the Cartesian sense) in the color space are equally distant perceptually.

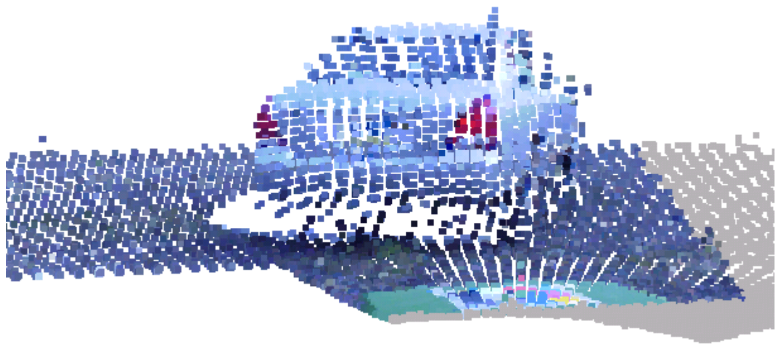


FIGURE 4.2. A 3-D model of a car built with our laser range finders. The calibration between the color cameras and the lasers allowed us to enhance the range measurements with color information.

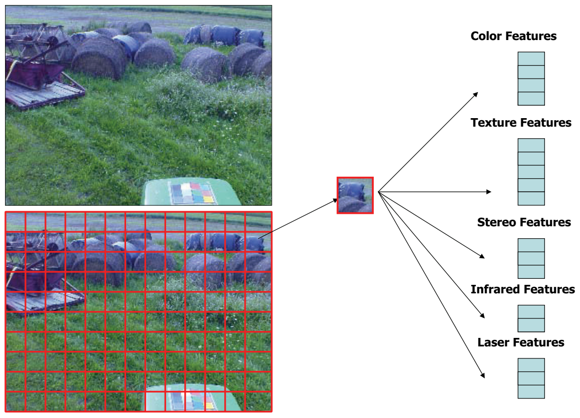


FIGURE 4.3. The feature vectors extracted from the data modalities available for each patch. The different feature vectors get concatenated in a larger feature vector x that represents the input to our learners and active learning methods.

and standard deviation over each patch, resulting in a six-dimensional feature vector. In all the experiments we present, we chose to ignore the mean value of the luminance channel (L), in order to make the color feature vector more robust to changes in illumination. Note that although a color calibration target is always visible in the images we recorded, we did not use any color constancy algorithm for the experimental results presented in this document.

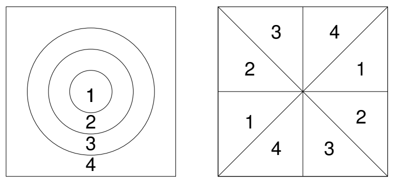


FIGURE 4.4. The division scheme that is applied to the FFT representations of our patches. Each patch is divided in concentric annular regions of increasing radii, corresponding to increasing frequencies (LEFT) and in wedges at different orientations, corresponding to the dominating orientations of the texture in the patch (RIGHT).

Texture: To extract texture features, we start from the frequency domain representation of each patch. The FFT image of the patch is split into several annular regions and wedges, as represented in Figure 4.4. The means and standard deviations of the absolute values of the responses in each region (ring or wedge) represent our texture features.

The intuition behind this representation came from the FFT domain representation of a bank of steered Gabor filters [**26**]. A Gabor filter looks like a Fourier basis element multiplied by a Gaussian, which means that the FFT representation of a Gabor filter is essentially a spot at a given frequency and orientation. As a result, our texture feature extraction scheme can be considered to be somewhat equivalent to convolving the original image patches with a bank of oriented Gabor filters. In our experiments we have used six scales and six orientations, which gives us a twenty-four dimensional texture feature vector for each patch.

Laser: As we have previously mentioned, each patch is associated with a list of 3-D points (laser range measurements) which project into it. The features we extract from these points are the average height in the vehicle frame, and the standard deviations in the vertical, forward and lateral directions. The motivation for using the standard deviations is that we believe they can be effective at discriminating between classes such as flat ground (small standard deviation in the vertical direction) and vegetation (large standard deviations in all directions).

Another option would be to compute the covariance matrix associated with the point cloud, and use its eigenvalues as indicators for different classes such as vegetation, linear or planar structures [**87**].

Stereo: An alternative to laser range finders is to use image pairs from both our cameras and obtain 3-D information through stereopsis. If the scene contains enough texture and its structure is such that the assumptions made by stereo vision hold[2], we will obtain for each patch a list of reconstructed 3-D points. We extract the same four features as in laser case: average height in the vehicle frame and standard deviations in the vertical, forward and lateral directions.

We have chosen not to use the features extracted from stereo in our experiments, mainly because the information they contain is much weaker than what we can extract from the laser range finders. Many of our data logs contain large amounts of tall weeds, which make the necessary step of matching between the two views quite difficult.

Infrared: The thermal camera we have integrated into our system can provide extremely valuable information for detecting humans and animals, or for discriminating between vegetation and man-made objects. Unfortunately, it is quite challenging to use thermal vision in an automated setting: the wide range of temperature variations that occur during a day make it hard to learn a thermal signature of a class of objects that is *generally* valid. A human body can be much warmer than the ground in early morning, and much cooler at noon. An image illustrating the disadvantages related to using infrared imagery for learning is presented in Figure 4.5.

We extract the mean and standard deviation of the pixel values from the IR image for each patch, which results in a two-dimensional feature vector. Although we have performed experiments using IR information, we choose not to present them here: most of our data was recorded at day time and in similar weather conditions, which makes IR a *much* stronger cue than it really is. Using IR in these conditions can make the obstacle

[2]This is generally not the case with tall vegetation such as weeds, for example.

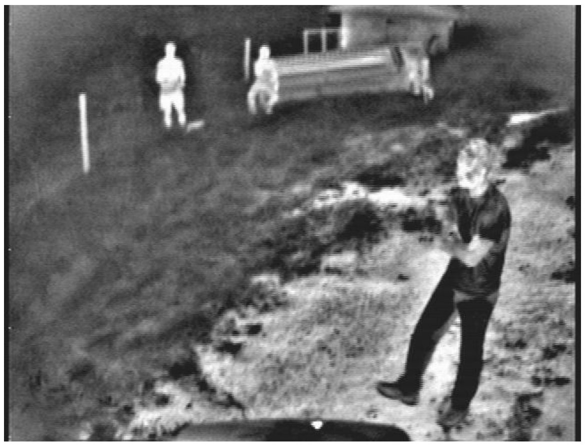


FIGURE 4.5. The worrisome limitations of thermal imaging. This image is a good reminder that infrared information needs to be used with *extreme* care. Within the same image, we have two humans in the background that over-saturate the image, and another human in the foreground who is under-saturated. The explanation of this very interesting image is the fact that the human in the foreground stayed in the shade for a few minutes before the image was taken, while the ones in the background did not.

detection problem seem misleadingly simple, especially since in most of our experiments the obstacles are humans or man-made objects that get warmer in the sun than vegetation.

When we refer to our experiments and indicate that they use “COLOR”, “COLOR + TEXTURE” or “COLOR + TEXTURE + LASER” features, we are referring to the fact that the learners and active learning methods are provided with the concatenation of the features extracted from those modalities as inputs.

The data sets on which we present results where collected at two different location: a natural meadow and a farm. In both cases, we have used the already present obstacles such as trees, fences, barns and other equipment and a set of obstacles that we placed in the environment in configurations of interest: humans in normal clothes or hunting jackets, poles of different thicknesses, solid objects placed lower than the surrounding vegetation, a ladder, a small water cooler and a green top of a tractor cabin. We have attempted to have obstacles whose detection is everywhere from relatively easy to extremely challenging even for a human. Although it is hard to describe the content of our data logs without displaying a large number of images, we will state our belief that if one could reliably detect most of the obstacles present in our data sets in the configuration where we placed them, that would be a great achievement for the outdoor navigation domain.

The next table presents statistics about the data sets we used, including the number of images in the log and the distance traveled. Entries that are grouped together (not separated by horizontal lines) represent data sets taken over the same area but driving on different paths, at slightly different times of the day. They are meant to be used at train/test data sets. For two of our data sets (`ApMdSpiral` and `FmVariousObs`) only one data set was available. However, given that we are measuring learning efficiency (how quickly do we learn as much as we would learn from labeling all the data?) and not generalization performance, overfitting is not a concern. To confirm this statement, we will show that both our best and worst results were achieved on these data sets, `ApMdSpiral` and `FmVariousObs` respectively.

Name	# Images	Distance traveled (m)
FmObsCourseNonPoles_01	436	372
FmObsCourseNonPoles_02	558	488
FmObsCourseAll_01	1018	749
FmObsCourseAll_02	673	481
ApMdSpiral	1393	768
FmDriveDown	2030	1627
FmDriveDownCloudy	1191	938
FmVariousObs	1096	543
ApHumanHard_01	222	106
ApHumanHard_02	249	119
ApParkWeedEasyHard_01	222	93
ApParkWeedEasyHard_02	190	104
ApParkObsInGrass_01	216	86
ApParkObsInGrass_02	216	94

Each data set has its own characteristics, and was collected for slightly different purposes. Bellow, we provide a very brief description of each set:

FmObsCourseNonPoles and FmObsCourseAll: The two pairs of data sets were collected on a field with weeds located at the farm. Other than a wall of trees visible occasionally, all obstacles were placed in this environment to form an "obstacle course" used for many other experiments performed with our platform. The course contained two parts: one with solid obstacles that where generally at the same level or lower than the surrounding weeds, and one with vertical poles of different thicknesses. As their names suggest, the FmObsCourseNonPoles did not contain any poles, while FmObsCourseAll covered the entire course. All the non-obstacle terrain present is covered with weeds or tall grass, while most of the obstacles only appear momentarily in the data logs and have relatively different signatures. This makes this data set well fit for our active learning experiments[3].

ApMdSpiral: This data set was collected at the meadow, and contains a longer drive, 90% of which is through vegetation that is between four and five feet tall. The terrain contained very few natural obstacles (only two small trees and a chain link fence) so we have built a short obstacle course (containing the same type of obstacles as FmObsCourseAll) to make the problem more interesting. The obstacle region is traversed only once, and represents a very small percentage of the entire data set. This makes the data set well fit for active learning experiments.

FmDriveDown: The pair of data sets contain the longest non-repeating drive that we could have recorded given our test sites. The data sets, recorded at the farm, contain a small portion of the obstacle course area, long stretches of farm dirt roads with nearby fences, a asphalt road stretch with occasional poles and mailboxes on the side, a few sheds and pieces of agricultural equipment plus a couple of negative obstacles. The terrain covered in the data set was not altered in any way, and it is representative for the types of objects that can be encountered on a farm. Because many of the obstacles (such as fences, or poles on the side of the road) are present in a large percentage of the data set, we expect the benefits that can be achieved by applying active learning to be small. This data set and the ones that follow were primarily collected for experiments not related to active learning.

FmVariousObs: This data set is a concatenation of several small data sets in which the vehicle starts 20-25m away from an obstacle at the farm and

[3]We will discuss later why a data set could be “unfit” for active learning experiments

drives up to it. The obstacles are mainly agricultural equipment, sheds, trees and fences, and their density is high.

ApHumanHard: This pair of data sets contains a short drive through the meadow area, in which a human wearing a hunting jacket is hidden in very tall vegetation. These data sets were collected for testing the limitations of our perception system: the detection problem is hard even for a human analyzing the data.

ApParkWeedEasyHard and ApParkObsInGrass: These two pairs of data sets were meant to be used as "toy" data sets for our initial active learning experiments. They contain small to medium height vegetation, and a series of artificial obstacles we placed in this environment. Their main advantage was their short length (approximately 200 images) which made it relatively easy to fully label them for repeated randomized experiments. One big disadvantage, however, is that a large number of obstacles are present, which, coupled with the short distance driven, makes the use of active learning techniques slightly futile: it is very unlikely that a randomly chosen image from these data sets does not contain an interesting obstacle.

Each data set was exhaustively manually labeled, so that in our randomized experiments we could query for the label of any image and receive it without any human interaction. We will discuss more details about these data sets and their effects on the active learning algorithms when we present our numerical results.

3. The Metrics

The fundamental quantity of interest in the active learning literature is the learning efficiency: we like know how effective an algorithm is at selecting data points (or data blocks, in our case) that lead to the best performance given a certain labeling effort. In general, authors choose a "one number" performance measure such as the error rate or the precision/recall break-even point (defined in **[41]**[4])

[4]The precision/recall break-even point is the value for which the precision and recall rates on a data set are equal. As Joachims points out in [**41**], there can be more or no break-even points for a data set.

and plot that value as the size of the training data set generated by active learning increases. Ideally, the authors also include an indication of the "maximally achievable performance", which is the performance that is achieved if *all* the unlabeled data is labeled. An active learning algorithm performs well when it results in generalization performance close to the maximally achievable one with much fewer labeled examples.

We follow this standard paradigm for presenting the results of our active learning experiments. However, we believe that for application domains with unbalanced data sets –such as ours– both the error rate and the precision/recall break-even point are poor performance measures which can be misleading. We choose to use the *area under the ROC curve (AUC)* score instead (see **[22]** for an excellent discussion of the properties of ROCs and AUCs).

Although Receiver Operating Characteristics (ROC) graphs have been used since the 1970s in the signal detection theory for presenting the tradeoffs between hit rates and false alarms, they were only relatively recently adopted on a large scale by the machine learning community**[22]**. Besides being an effective representation method providing a large amount of information in a single graph, ROCs have the important property that they are insensitive to data sets that have unbalanced class priors.

ROCs are primarily a method for representing the performance of a binary classifier[5]. If we assume that the classifier has to discriminate between negative and positive examples, an ROC graph is a plot of the true positive rate (TP_{rate}) versus the false negative rate (FN_{rate}).

Following the notation used in Figure 4.6, the TP_{rate} is defined as the ratio $\frac{TP}{P}$, which is the percentage of the positive examples in the data set that are correctly identified as positives by the classifier. The FP_{rate} is the ratio $\frac{FP}{N}$, which is the percentage of the negative examples in the data set which are misclassified as positives. In principle, one could generate an ROC curve by varying in very small steps the decision threshold between the two classes, and measuring the rates for each

[5]See **[22]** for a discussion of certain extensions of ROCs to the multi-class case.

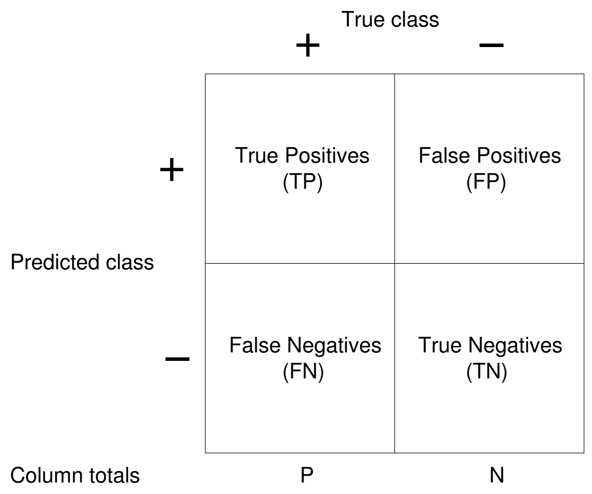


FIGURE 4.6. The definition of true and false positives, along with true and false negatives for a binary classifier discriminating between the "+" and the "-" classes. P, the number of positive examples in the data set, is the sum of true positives and false negatives. N, the number of negative examples in the data set, is the sum of false positives and true negatives. This figure follows closely the one presented by Fawcett in [**22**].

threshold setting. Fawcett [**22**] describes a far more efficient method for generating ROC curves, which is used in our experiments.

The obvious advantage in using ROCs instead of error rates for reporting classification performance, is that an entire curve is presented, instead of just one point on it which happens to correspond to the decision threshold used to compute classification rates. This is important, because the ROC curves of two different classifiers can very well intersect, which means that depending on the setting of the decision threshold, the ordering of the classifier performance can change. Considering that most classifiers are not correctly calibrated (i.e. they do not produce correct posterior probabilities), it is unclear, a priori, what decision threshold should be used. If ROC curves are employed, one can choose a setting based on the class misclassification costs and the desired operating range.

However, the most important advantage in using ROC is not the obvious one. Precision/recall curves[6], a performance comparison tool widely used in retrieval tasks, also present a multitude of operating ranges, but lack a key property that

[6]"Precision" is defined as $\frac{TP}{TP+FP}$, the percentage of retrieved examples which are correct. "Recall" is defined as $\frac{TP}{P}$, the percentage of correct examples existing in the data set which were retrieved.

ROCs have: the independence from *class skew*, the ratio of negative and positive examples in the data set.

To understand why ROC's immunity to class skew is important, consider the following example: assume we have a very simple classifier which, when presented with a training set containing instances from the "+" and "-" classes, identifies the most frequently encountered label, and always uses it for prediction, completely ignoring any input features. Let us assume that in the training set, there are more negative examples than positives. When presented with a test set, the classifier will always predict "-" labels. Depending on the percentage of negative examples in the test data set, the error rate of the classifier can vary drastically: for a balanced data set, the error rate will be 50%. However, if the negative examples represent 90% of the test set, the correct classification rate achieved will be a very misleading 90%. Thus, by adding more examples from one class to the test data set, one can dramatically change the performance estimates. Furthermore, even if the classifier is slightly "smarter" and does not ignore the input features, it is not clear what error rate will represent "good" performance. One would have to carefully analyze the test set before deciding whether a 95% error rate is a significant achievement or not.

Since the TP_{rate} and the FP_{rate} used to generate an ROC plot are computed using entries in the same vertical column of Figure 4.6, they are immune to changing the ratio of positives and negatives in the test set: the ROC representing the performance of a classifier on a perfectly balanced data set is identical to the ROC generated using the same classifier on a data set with 99% of its examples from one class. Since the formulas used to compute precision and recall estimates use entries from both vertical columns in Figure 4.6, precision/recall curves lack this property.

In our domain, the data sets we collect are generally very skewed, and to an arbitrary degree. When collecting a data set with obstacle/non-obstacle examples one could, for example, drive straight to the obstacle or they could drive many times in a circle in a non-obstacle area and then approach the obstacle. We would not want our performance estimates to depend on such factors.

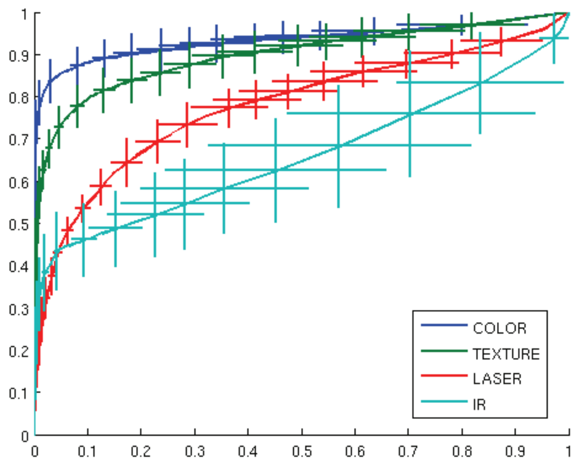


FIGURE 4.7. The ROC curves describing the performance of a classifier the same data set, using feature vectors extracted from different sensing modalities. The confidence "crosses" in the figure are generated using the threshold-averaging method described in [**22**]. The closer the ROC of a classifier is to the upper-left corner of the plot, the better its performance.

We have just argued that ROC curves are a better way of representing performance than error rates or precision/recall break-even points. However, an ROC is a *vector* of numbers, which cannot be easily plotted and interpreted in the context of active learning experimentation. To have an informative and concise image of how well an active learning algorithm performs, we need a "one-number" summary of an ROC curve, which would allow us to generate the typical active learning performance graphs we described at the beginning of this section. Fortunately, a well studied and meaningful scalar summary of an ROC curve exists: it is the area under the ROC curve, the AUC score.

For any ROC curve, the AUC score is given by the area of the region delimited by the ROC curve and the bottom and right edges of the unit square. If we analyze Figure 4.7 it is clear that the closer the ROC of a classifier is to the upper-left corner of the plot, the better its performance: for any point M in the TP_{rate}-FP_{rate} space, a point contained in the rectangle described by the edges of the unit square, the upper-left corner and the point M corresponds to better performance. If the ROC curve corresponding to a classifier A always dominates the curve for a classifier B, then the classifier A is always preferable to B. The AUC score for the classifier A will also be higher than the AUC for B.

Since the AUC is a portion of the unit square area, it can take values in the $[0, 1]$ interval. However, it can be shown that the ROC curve of a completely random classifier is (for a large enough sample) a 45 degree line from $(0, 0)$ to $(1, 1)$. As a result, no reasonable classifier should have an AUC smaller than 0.5: if it does, one can swap the labels of the two classes and obtain an ROC that is symmetric with respect to the $(0, 0) - (1, 1)$ diagonal of the unit square **[22]**.

In 1982, Hanley and McNeil **[32]** have shown that the AUC score has a well defined statistical meaning: it is the probability that a classifier discriminating between positive and negative examples will produce a higher output for a randomly selected positive example than for a randomly selected negative example. The AUC can thus be used as a meaningful scalar descriptor of an ROC curve.

To generate active learning performance plots, we will proceed in the following manner:

1. Prepare a pool U of unlabeled data from which to select images for labeling
2. Prepare a (preferably separate) labeled test data set L_{test}
3. Initialize the active learning algorithm and start the iterative process of selecting images for labeling
4. Each time an new image is selected for labeling, it is added to set of labeled data available for training L_{train}
5. Each time the data set L_{train} is augmented, a classifier is trained using its content, and the AUC score on the L_{test} data set is computed
6. The sequence of AUC scores obtained through the active learning iterations is the "learning curve" for the algorithm used.

Since we compare active learning algorithms based on their learning curves, it is very important to have a confidence interval for each AUC estimate used to generate the learning curves[7]. To obtain confidence intervals for each AUC score, we need to perform repeated randomized trials with each experiment. Randomization is inserted in our experiments through the initialization process, as we will show

[7]Ideally, we would want to have estimate *bands* rather than point-wise intervals.

in Section 4. Once several AUC score estimates are available for each size of the training data set, we use their mean to plot the learning curve, and the standard deviation for displaying confidence bars.

The errors bar we plot around the mean AUC score enclose an interval of $\pm 1.64\sigma$, which is the correct length to use in order to claim 95% significance of the ordering of the estimates when the error bars do not overlap [**50**]. We will display symmetric bars which is slightly wrong, given that the AUC score is bounded by $[0, 1]$[8].

It is important to realize that for each number of labeled images on our plot, the error bars we display represent confidence intervals for the mean estimate, and not for one random run. Regardless of the intrinsic variance V_{AL} of an active learning algorithm, with a very large number of experiments the confidence interval for the mean AUC score would eventually shrink to 0, since the variance of the mean is $\frac{V_{AL}}{N}$, where N is the number of randomized runs of the algorithm.

While using the (smaller) standard error for the mean estimate for our plots instead of the one for an individual run is correct from a statistical point of view, for certain applications knowing the standard deviation of one run is very important. As Figure 4.8 suggests, sometimes we might prefer an algorithm with lower average performance but smaller variance to one with higher average performance and higher variance.

Since all of our plots represent the averages of at least ten randomized runs, the reader could mentally multiply the error bars we display by $\sqrt{10} = 3.1623$ in order to estimate the significance of the ordering of the AUC scores of one random run of each algorithm.

[8]We have considered using the Hoeffding bound, but unfortunately the bound is too lose give our sample size of 10.

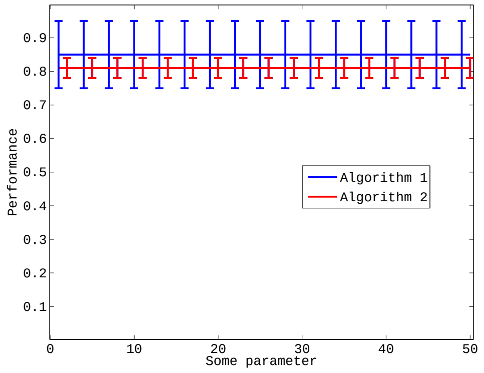


FIGURE 4.8. The average performance of two hypothetical algorithms, A1 and A2, along with the confidence interval for one random run of each algorithm. Although the algorithm 1 will do better on average, it can also generate worse performance than algorithm 2. Depending on the cost function for a specific application, the higher worse-case performance of the algorithm 2 might make it preferable to algorithm 1.

4. The Experimental Procedure

To answer the questions described in Section 1 while reducing the chances of spurious observations, we have performed repeated experiments while varying four major factors that could affect the learning efficiency:

1. The initialization method
2. The data selection algorithm
3. The data set
4. The types of sensing modalities (e.g. color, color+texture, etc.)

As we have already argued, the initialization procedure can have a major impact on the performance of many active learning algorithms. Using an initialization data set constructed manually by a human expert is a solution that has been reported in the active learning literature to produce good results, but is impractical once we consider applying algorithms to large data sets. We have chosen to compare the performance of three initialization procedures: manual, random and UDF-based data selection. The details involved in each initialization procedure are described bellow.

MANUAL: Although manual selection is impractical for large data sets, it is still a "golden standard" in the active learning community. As a result, we have decided to measure the effectiveness of manual initialization, in order to estimate the relative performance of the other methods that have better scaling properties.

In order to perform repeated randomized experiments without human interaction while still using manual initialization, we have extracted from each data set a list of the images that would be considered by a human expert to be appropriate for initialization. Since the task of interest for our experiments is obstacle detection, we have manually identified and listed those scenes from each data set that contain close-up views of the obstacles, and interesting examples of non-obstacles if they existed.

At runtime, the MANUAL selection method gets to randomly sample without replacement images from the list created by the human expert. The number of images sampled depends on the specific experiment performed.

RANDOM: As expected, random image selection is the immediate solution for initialization when a data set is too large for using manual selection. The RANDOM method samples images randomly from the entire unlabeled data set available, without replacement. As our experiments will show, random initialization can lead to poor performance in data sets with severely unbalanced class priors.

UDF: We have described the Unlabeled Data Filtering algorithm in Section 1 of Chapter 3 on page 37. The algorithm works by trying to provide good coverage of the feature space, and it has been developed as a solution to the random initialization problems.

Unlabeled Data Filtering itself requires some initialization data in order to proceed with estimating a probability density over the feature space. However, since its goal is to obtain data in most regions of the feature space, UDF is almost indifferent to the specific data used for initialization. In our experiments we randomly pick an image from the unlabeled data set, and proceed with the algorithm as previously described.

In choosing which data selection algorithms to consider in our experiments, we have followed the same reasoning as for the initialization methods: MANUAL selection is likely to lead to good performance but is impractical, while RANDOM selection is an immediate solution that can lead to poor performance in certain cases. Against these two standard selection methods, we have tested two active learning methods that were most successful in our early experimentation stage: the Query-by-Bagging and the Unlabeled Data Filtering algorithms. In all of our plots we have also included a horizontal line labeled “MAX INFO”, which corresponds to the performance achieved by the classifier if all the data available in the unlabeled data set was labeled. Ideally, we would want our data selection methods to get very close to the MAX INFO line as fast as possible, since this would indicate a high labeling efficiency.

To answer the most important question in Section 1 regarding the applicability of active learning techniques in our domain, we have compared the MANUAL, RANDOM, QBBAG and UDF selection methods on all our fully labeled data sets, using three sets of input features: Color, (Color + Texture), and (Color + Texture + Laser). The specific features extracted from each sensing modality were described in Section 2 of this chapter. By considering these three sets of features, we intended to vary the input dimensionality from five dimensions (Color) to thirty-three (Color + Texture + Laser) in order to verify the performance of the Unlabeled Data Filtering algorithm in high-dimensional feature spaces.

In the beginning of this section, we have stated that we want to understand the sensitivity of our algorithms to the various parameter settings. To verify that our results are repeatable and robust to these settings, we have explored the sensitivity of our algorithms to each one of the important parameters. Furthermore, we would like to stress the fact that *all* the experimental results presented in this thesis[9] were obtained using *exactly* the same parameter settings. We have not changed parameters based on the set of features, the data set or the specific active learning

[9]with the exception of the sensitivity analysis, of course

algorithm used. We believe this approach to measuring performance is an important pre-condition for being able to estimate the importance of the techniques we proposed in real-world applications.

In the following subsections we will discuss the setting of three parameters that are important for the experiments presented in this thesis:

- the number of patches from each image over which we accumulate the interest scores
- the number of classifiers in the Query-by-Bagging committee
- the number of dimensions of the space in which the original data is projected with PCA for the Unlabeled Data Filtering algorithm.

4.1. Choosing the number of patches used for accumulation

As we have argued in the section describing the data block constraint, we expect the performance of active learning algorithms to be limited when the accumulation of interest scores for a block happens over either too many or too few image patches. These two effects are illustrated in Figure 4.9, in which we plot the mean AUC score obtained by the QBBAG algorithm on color and texture features (initialized randomly) as we increase the number of patches over which the image interest score is accumulated.

The left plot confirms our expectation that when accumulating over a very small number of patches, the active learning performance can degrade, due the the artifacts that are present in our data. Using four or eight image patches for accumulating the interest score leads to better performance than when using one or two patches, both in terms of mean AUC score and its standard deviation.

The right plot explores the opposite extreme, when the accumulation step takes place over a large percentage of all the image patches present in the image. The performance obtained when accumulating over eight or even thirty image patches is superior to the results obtained when accumulating over two hundred

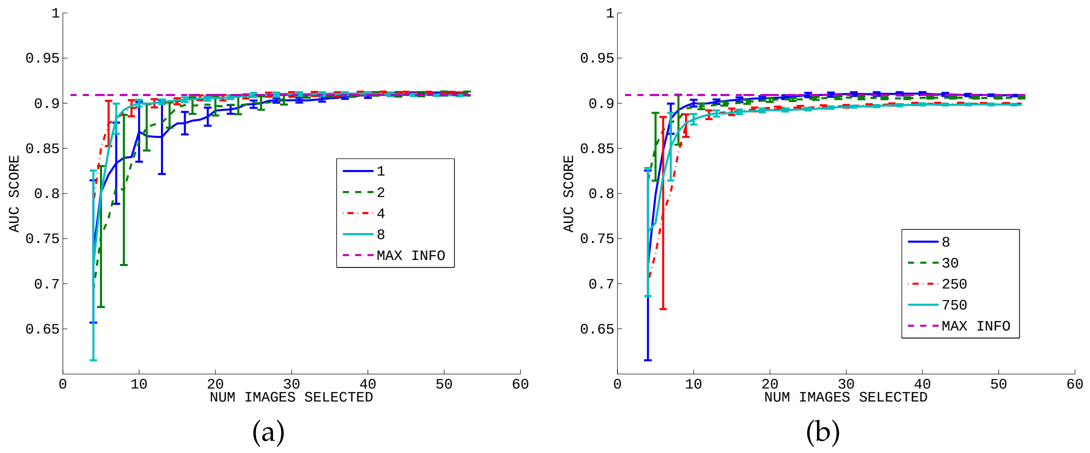


FIGURE 4.9. The effect of accumulating interest scores over different numbers of image patches. The algorithm used to generate these graphs was QBBAG with RANDOM initialization, using COLOR+TEXTURE features. As expected, using a very small number of patches (one or two) leads to worse performance since the algorithm is more likely to be affected by artifacts in the data (LEFT). As the number of patches used for accumulation increases dramatically, the active learning performance becomes slightly lower, as algorithms are less likely to select for labeling images that contains *small* interesting obstacles (RIGHT).

and fifty and seven hundred and fifty patches. The difference in performance is relatively small, but statistically significant even for an individual run. These results suggest that when combining the interest scores over a large portion of the image, we are biasing the active learning algorithm against considering images with patterns that are "interesting" but only occupy a small portion of the image.

Our experiments suggest that for the simple accumulation methods which does not impose spatial coherency, averaging interest scores over a number of four to roughly fifty image patches produces very similar results. As a result, we have performed all our active learning experiments with accumulation over the eight patches with the highest interest scores in each image.

4.2. Choosing the number of classifiers in the Bagging committee

One of the design choices expected to affect the performance of the Query-by-Bagging algorithm is the size of the classifier pool that is used to estimate the

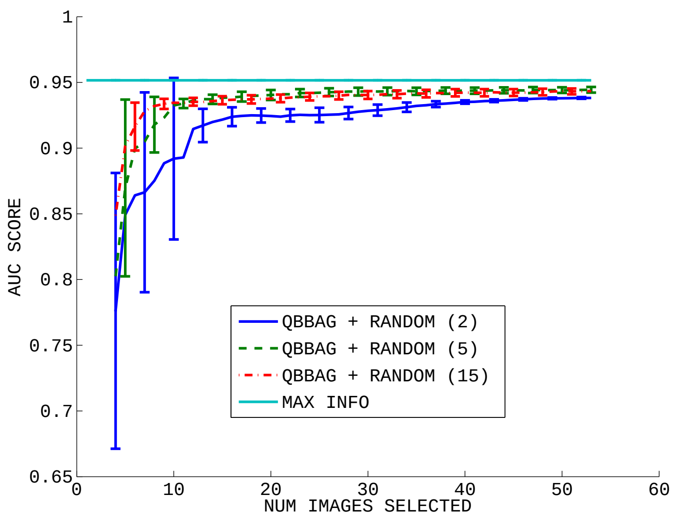


FIGURE 4.10. The effect of the size of the Bagging pool on the performance of the QBBAG algorithm. The larger pool of classifiers (15) results in performance that is superior in terms of the mean AUC score and the standard deviation of the estimate.

expected reduction in size of the version space. The original QBC paper [**81**] argues that a pool with as few as two classifiers should be sufficient, while Abe and Mamitsuka [**1**] choose to use a pool of twenty classifiers instead.

Using the KL-divergence-to-the-mean disagreement method we have described, we performed experiments varying the size of the classifier pool from two to fifteen, using color and texture features on the `FmObsCourseNonPoles` data set. The results are presented in Figure 4.10, and they agree with our intuition that a larger pool would allow a more precise estimation of the expected reduction in the version space size. The plot indicates that QBBAG version using a committee of fifteen hypotheses leads to performance that is superior in terms of mean and standard deviation to the results obtained using a smaller committee of two. All of our other experiments involving Query-by-Bagging employed committees of fifteen classifiers.

4.3. Choosing the dimensionality of the UDF space

The Unlabeled Data Filtering algorithm relies on repeatedly estimating a probability density function over the feature space. However, since our classifiers often

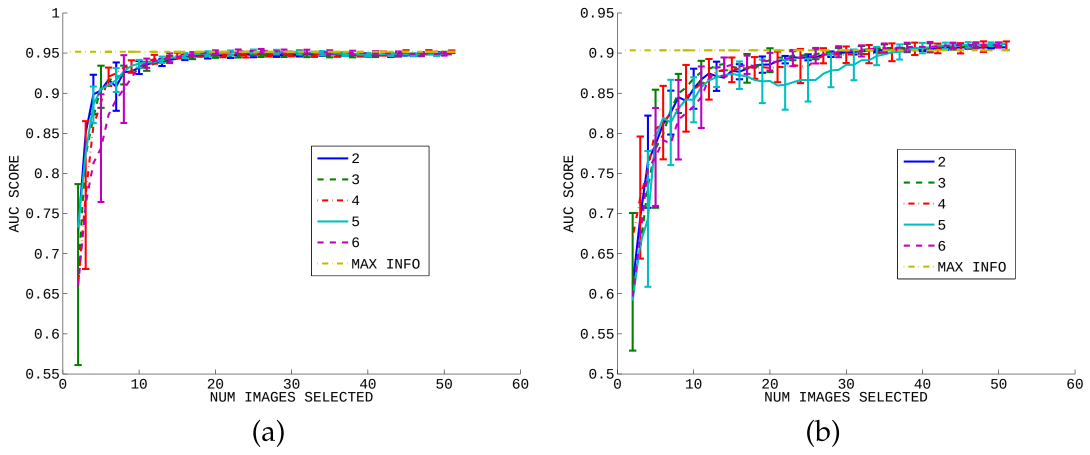


FIGURE 4.11. The effect of the number of dimensions retained when compressing data for the UDF algorithm. When using (Color + Texture) features (LEFT) there is no significant difference in performance when varying the dimensionality from two to six. However, when using the less informative Texture feature set, more compression resulted in slightly tighter confidence intervals at similar average AUC scores.

work in very high-dimensional spaces, it is necessary to perform a dimensionality reduction step in order to avoid the "curse of dimensionality" that affects all density estimation methods, including the Kernel Density Estimation (KDE) method UDF uses. In our experiments, we have chosen the well known Principal Component Analysis (PCA) method as our dimensionality reduction algorithm.

When choosing the dimensionality of the space to which we want to linearly project our data using PCA, there is a trade-off between retaining too many dimensions which would hamper the performance of KDE, and compressing the data too much which can result in significant information loss.

To estimate the sensitivity of the UDF algorithm to the dimensionality of the destination space, we have performed experiments using Texture and Color + Texture features on the `FmObsCourseNonPoles` data set. The performance of the active learning algorithm when the dimensionality is varied from two to six is displayed in Figure 4.11.

Interestingly, the results were quite different when using the two sets of features. For the (Color + Texture) feature set (left), there is no significant difference in performance when the dimensionality of the space is varying from two to six.

For the Texture feature set (right), the differences are still small but the standard deviation of the results obtained when the data is projected to two dimensions is the smallest.

We hypothesize that for the data set used, color information is a strong cue and that none of the first five or six PCA dimensions is dominated by noise. At the same time, the large number of patches that we use for estimating the probability density function seems sufficient for performing KDE in this relatively high-dimensional space. For the texture feature set, we believe that only the first two or three PCA dimensions contain relevant information, and that efforts spent into populating additional dimensions do not result in additional information for classification.

This explanation is only a weak hypothesis, and additional experimentation is necessary for a more qualified explanation of our observation. Given that the main focus of this thesis is much broader, we postpone this additional investigation for future work. Based on our experimental observations, we have chosen to use a two-dimensional space for performing the kernel density estimation step of the UDF algorithm.

4.4. The Base Learner: Logistic Regression

To be able to perform the type of experiments that were necessary for our goals, we needed a classifier that is fast to train and apply and that can handle data noise and has good convergence. We would also like it to have outputs that have a probabilistic interpretation (such as class posteriors).

Although we have initially performed a small number of experiments using neural networks, all the results we present in this thesis were obtained using logistic regression, a standard classification algorithm [**7, 35**] that has the properties we are interested in. Logistic regression is a linear classifier which, despite –or perhaps due to– its simplicity, consistently ranks among the top learning algorithms for a large variety of real-world classification problems. Fitting logistic regression models is achieved by solving a convex optimization problem, typically using the iteratively reweighted least-squares algorithm (IRLS, [**35, 60**]). Komarek [**43**] proposed several techniques and modifications for making logistic regression successfully scale to extremely large data sets.

5. Results

5.1. Active Learning vs. Random and Manual Data Selection

The experiments presented in this section are meant to determine if the active learning methods we considered are at all effective at reducing the data labeling requirements for learning problems that are representative for our domain. It is important to determine if reasoning about the distribution of the unlabeled data using active learning techniques leads to *any* substantial advantage over the default RANDOM selection procedure.

In case of a positive answer, it is also interesting to compare the performance of our active learning techniques with the MANUAL selection procedure. Although we have argued that manual selection is impractical for large data sets, it would be reassuring to know that active learning is comparable in efficiency to a human expert, if a sufficient amount of time was available for the expert to analyze all the available unlabeled data.

As a first qualitative evaluation, we have selected 50 images from the `ApMdSpiral` data set using random sampling, and 50 images using UDF on color and texture features. We have assembled the results in large image mosaics, which are presented in Figure 4.12 (RANDOM) and 4.13 (UDF). In the images composing the UDF mosaic, we have indicated in red the image patches that generated the 8 highest interest scores, and who are thus "responsible" for the selection of the image. The images are ordered row-wise, from left to right.

The image mosaics illustrate the intuition that led to the development of the UDF algorithm. The vast majority of the images in the `ApMdSpiral` data set contain only vegetation, while most obstacles are concentrated at the very beginning of the data log. This is a case where the class priors are very unbalanced, which could cause random sampling to ignore important information about the rare class.

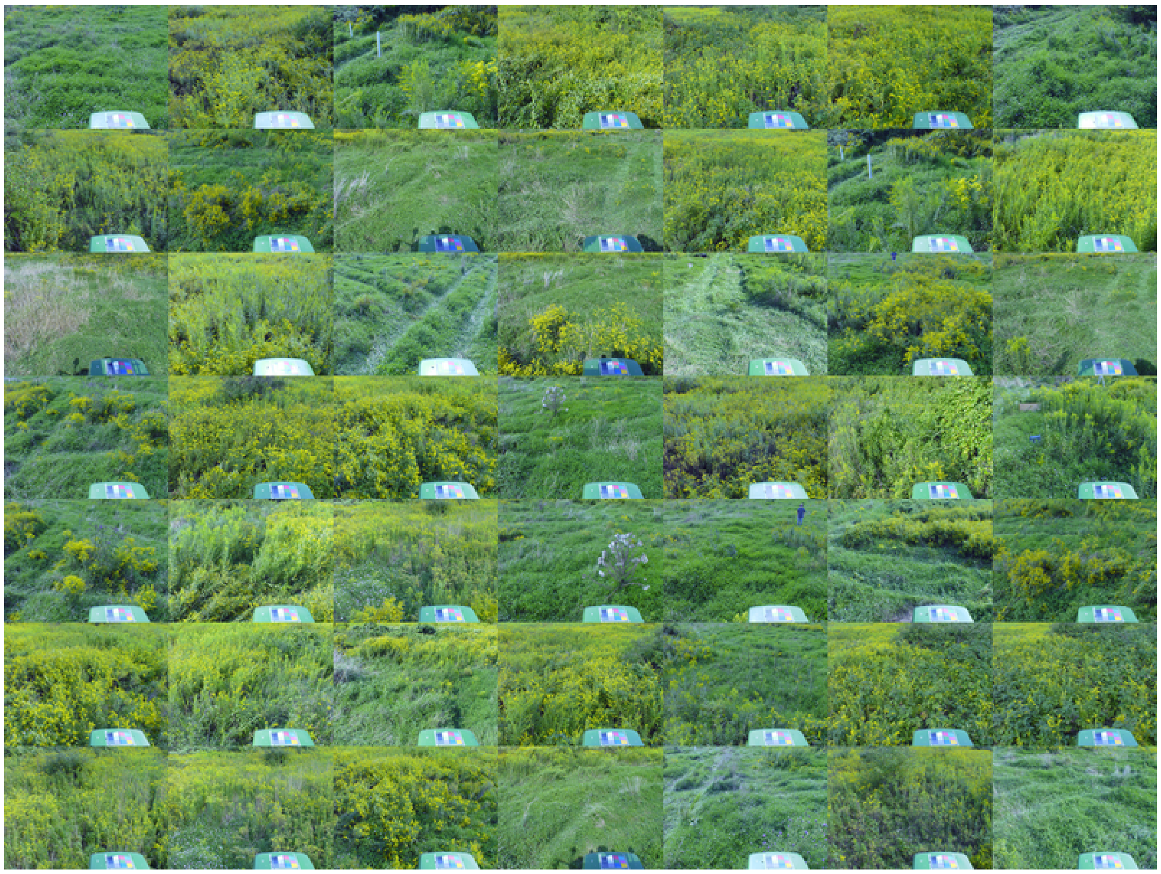


FIGURE 4.12. Images selected randomly from the ApMdSpiral data set. Since class priors are extremely unbalanced, only a very small percentage of the images (3, 13, 28 and 33) contain interesting obstacles. We would expect a classifier trained using this data to be a relatively poor obstacle detector.

Figure 4.12 shows that the phenomenon indeed occurs. Out of the first 49 images selected in the our first RANDOM run, only four images contain obstacles, two of them repeated.

The mosaic in Figure 4.13 contains a much higher percentage of images of obstacles. Unlabeled Data Filtering selected images of poles, people, deep shadows, a cart, pots hidden in vegetation and a dirt pile. The red patches displayed in each image confirm that the images were indeed selected because of the pattern that we, humans, would consider interesting in those scenes. What is important to realize,

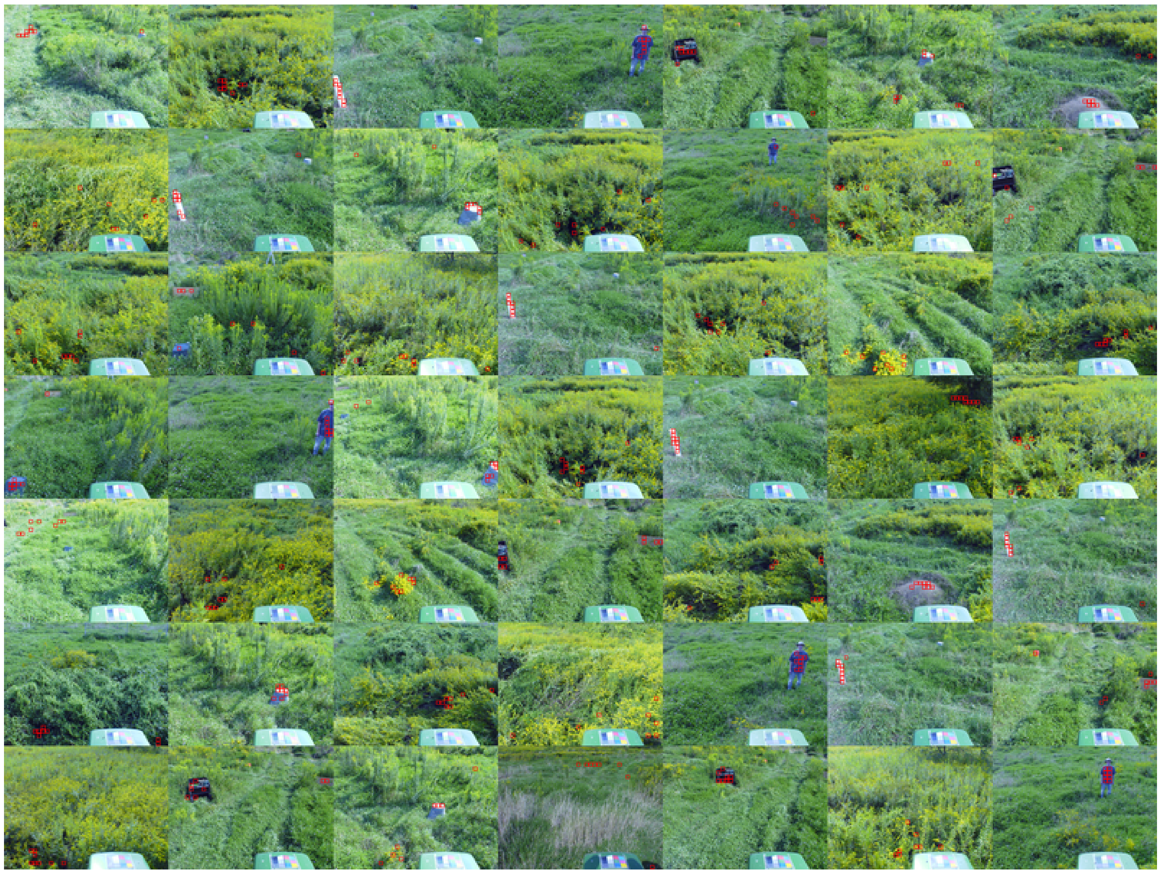


FIGURE 4.13. Images selected by the UDF algorithm from the ApMdSpiral data set, using COLOR + TEXTURE features. The percentage of images containing interesting obstacles is much higher than in the data set selected by random sampling. The red squares, indicating the image patches used to compute the interest score of each image, correspond to our human intuition of what is "unusual" about an image; however, UDF selected the images in this mosaic without having any concept of "obstacle". We would expect a classifier trained using this data to be a relatively good obstacle detector.

however, is that UDF selected all the images *without knowing anything about the concept of obstacles*: the only information used to select the images in Figure 4.13 was the distribution of the image patches in the feature space.

We would expect an obstacle detector trained using the data selected by the UDF algorithm to perform much better than a detector trained using the randomly selected images. Our quantitative results (see for example Figure 4.21) show that this is indeed the case.

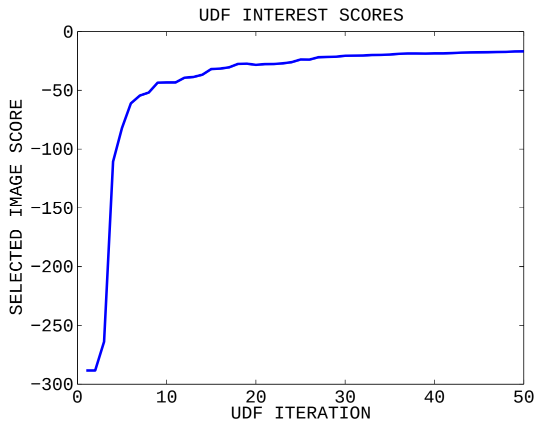


FIGURE 4.14. The evolution of the image interest score as more images from the ApMdSpiral data set are selected using UDF. The curve flattens after 10-15 images, which suggests that the feature space (its two-dimensional projection) is well covered at that point. This curve can be used as a heuristic for choosing when to stop labeling additional images. Figure 4.21 confirms that indeed, for this data set, UDF gets very close to is peak performance after observing 10-15 labeled images.

Figure 4.14 illustrates the evolution of the interest score for the images selected by UDF, as more images are added to the selected data set. We notice that the curve flattens after 10-15 images, which suggests that the feature space (more precisely its two-dimensional PCA projection) is relatively well covered at that point. This curve can be used as a heuristic for choosing when to stop labeling additional images: Figure 4.21 indicates that indeed, the classifier gets very close to its maximal performance after observing 10-15 labeled images.

Based on this curve, we have trained two versions of logistic regression classifiers using color and texture features. One of the classifiers was trained using the first 10 images selected randomly, while the other was trained on the first 10 images selected by UDF. An example of the outputs produced by the two classifiers are displayed in Figure 4.15. Not surprisingly, the classifier trained using random selection completely fails to detect the black cart in the image, since its training set did not contain any examples of black texture-less objects that were obstacles.

Our preliminary quantitative experiments indicated that the Query-by-Bagging algorithm performs similarly to Co-Testing, when using (Color + Texture + Laser)

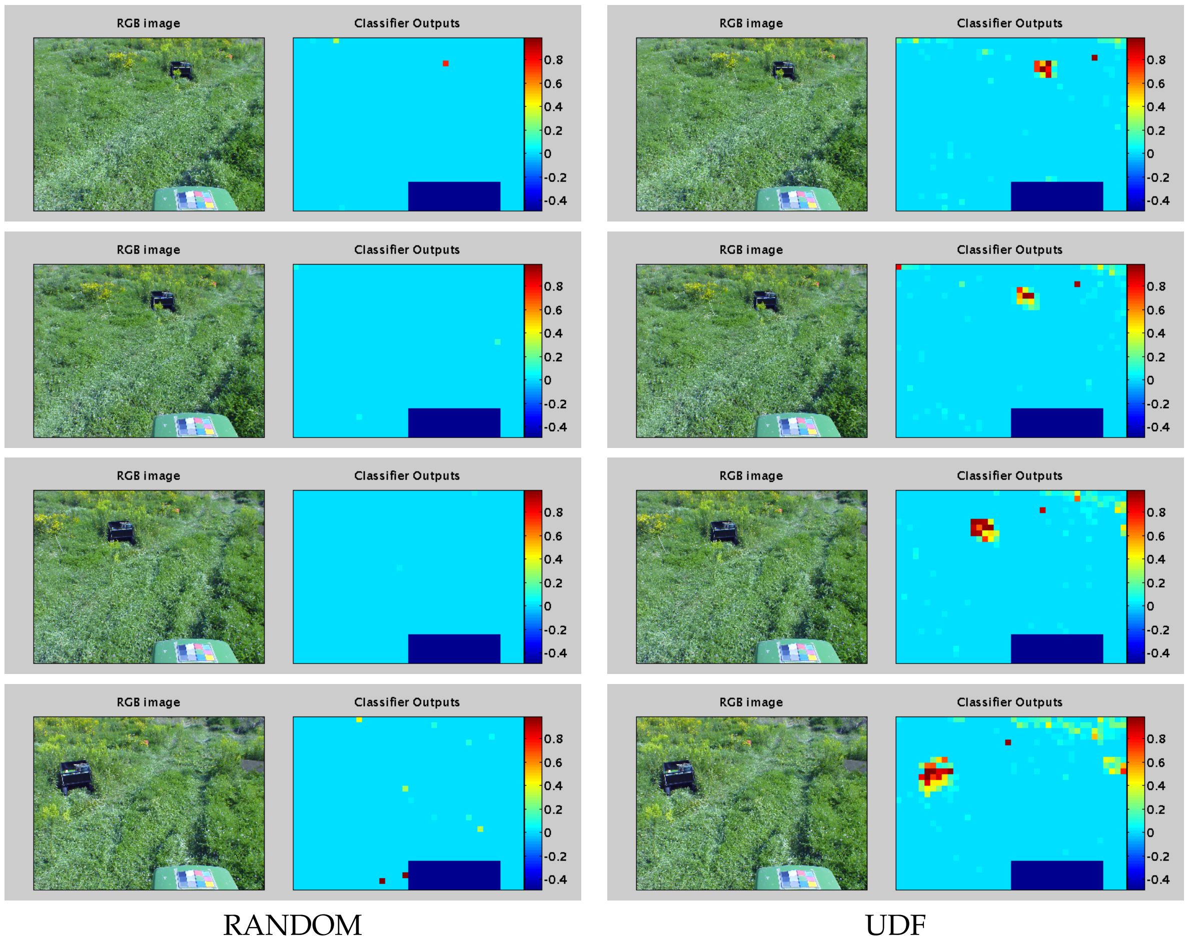
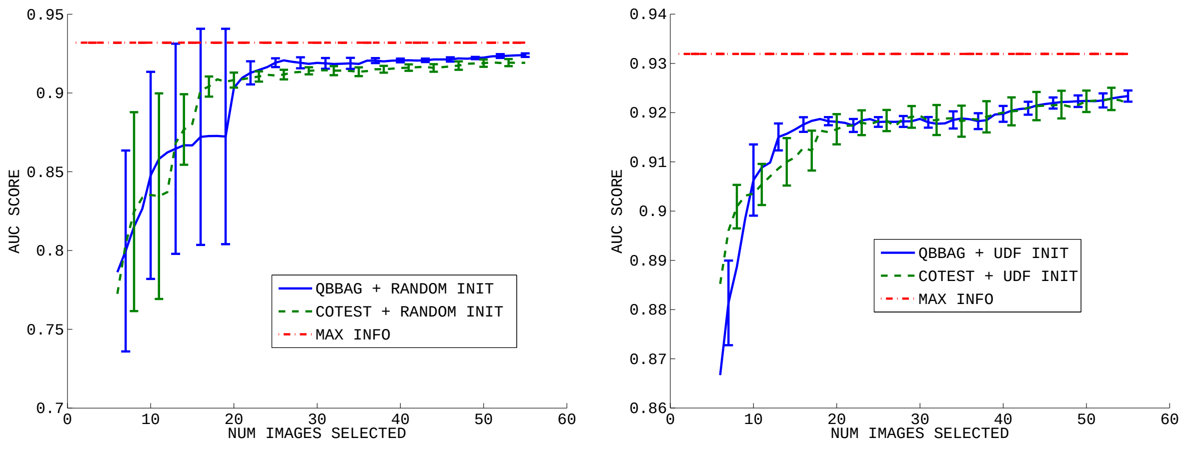


FIGURE 4.15. The classification outputs produced by a classifier trained using the first 10 images selected randomly (LEFT), and the first 10 images selected by UDF (RIGHT). The classifier is using color and texture features. The color coded outputs represent the probability of the "obstacle" class. The classifier trained using random selection completely fails to detect the black cart in the image, since its training set did not contain any examples of black texture-less objects that were obstacles.

features for QBBAG and the Color, Texture and Laser feature sets as the three Co-Testing views (see Figure 4.16 for an example). The performance of the Uncertainty Sampling algorithm was generally worse. Since the Co-Testing algorithm is somewhat limiting in the experiments we could do because of its requirement of having multiple independent views of the data, we have chosen to use Query-by-Bagging as the "standard" active learning algorithm to compare against.

Figures 4.17 through 4.21 present our quantitative results on the five major data sets we considered. The plots compare the performance of RANDOM and



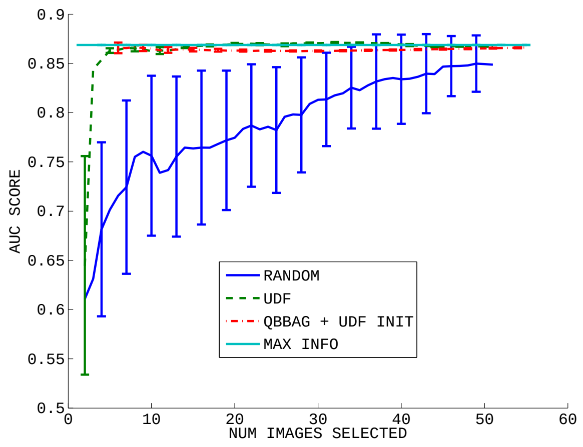
Random Initialization UDF Initialization

FIGURE 4.16. Comparing the performance of Query-by-Bagging to Co-Testing on the ApMdSpiral data set, using RANDOM (LEFT) and UDF (RIGHT) initialization. Both algorithms use the KL-divergence-to-the-mean diversity measure to estimate the disagreement between the committee members for QBBAG and the different views of the data for CO-TESTING.

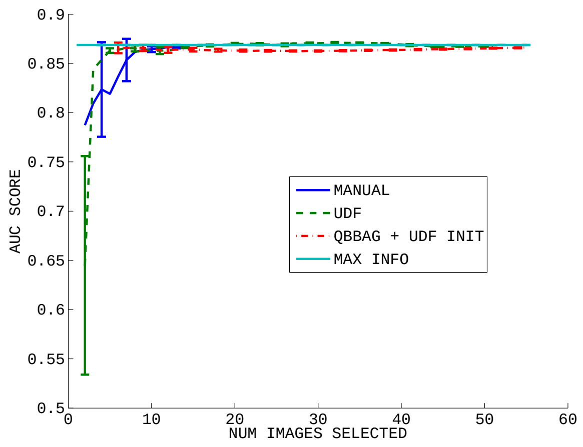
MANUAL selection with that of UDF and Query-by-Bagging initialized with UDF, on Color, (Color + Texture) and (Color + Texture + Laser) features. The left column in each one of the plots compares the two active learning methods with RANDOM selection, and the right column compares them to MANUAL selection.

In Figure 4.17, the performance of both active learning algorithms is dramatically superior to that of RANDOM selection for Color and (Color + Texture) features (left column, first two rows). Both algorithms get extremely close to the MAX INFO performance very fast, in about 10 images. Their confidence intervals are tight, which means that the good results are consistently achieved. In the right column, we notice that the performance of the active learning methods is essentially identical to that of MANUAL selection method, indicating that they are as effective as a human expert analyzing the entire data set.

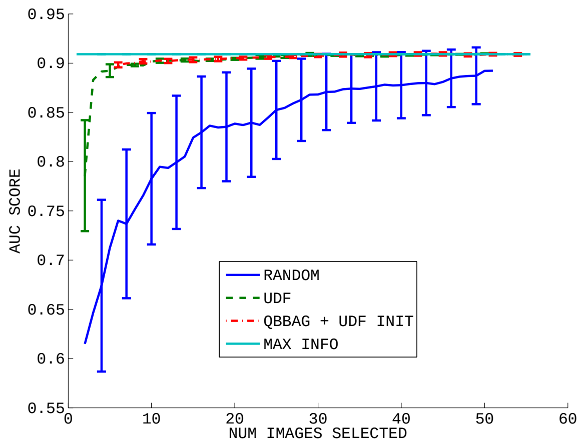
For the (Color + Texture + Laser) feature set, the performance of the QBBAG algorithm initialized with UDF was as good as for the other two feature sets, and much superior to RANDOM selection. UDF initially outperforms RANDOM selection (up to the first 10 images) and then only grows slowly, eventually being overtaken after 30 images. We believe that the reason is that due to the nature of



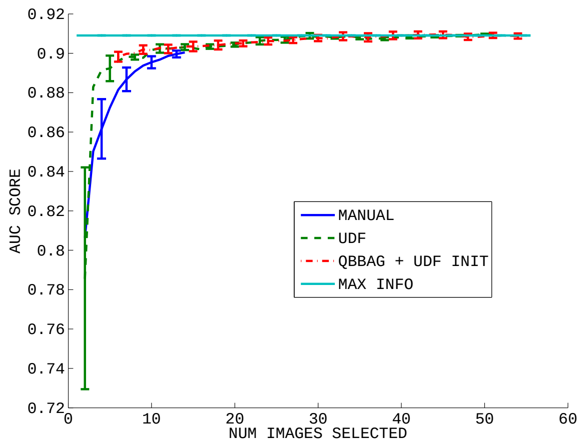
COLOR (AL vs. Random)



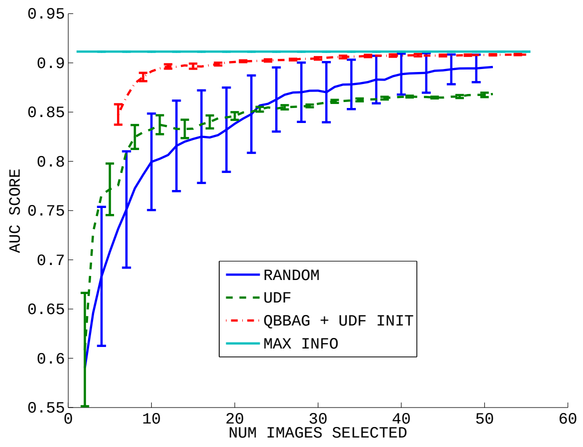
COLOR (AL vs. Manual)



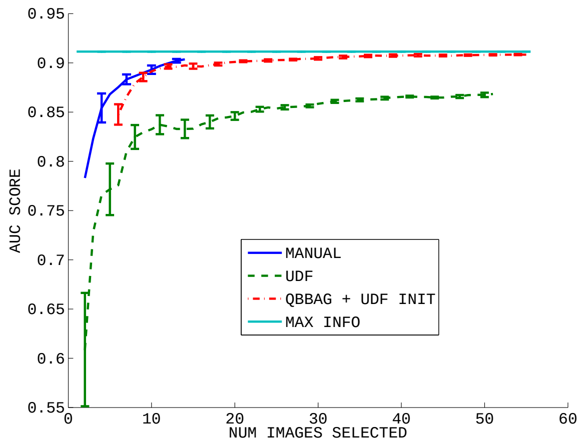
COLOR + TEXTURE (AL vs. Random)



COLOR + TEXTURE (AL vs. Manual)



COL + TEX + LASER (AL vs. Random)



COL + TEX + LASER (AL vs. Manual)

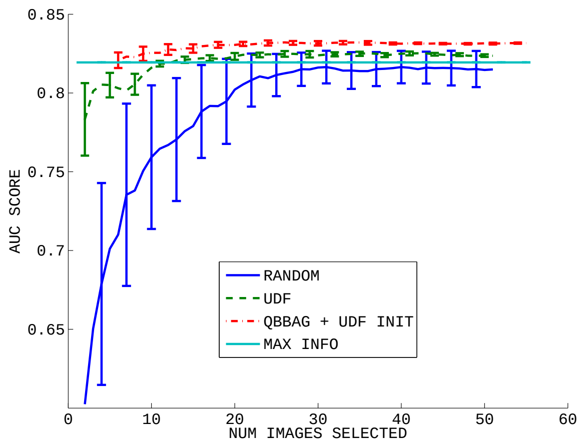
FIGURE 4.17. Results on the FmObsCourseNonPoles data set.

the obstacles, the laser features bring almost no additional information for this data set. As a result, after the initial 10 images, UDF attempts to account for the variation in the laser data which does not lead to an increase in classification performance on the test set.

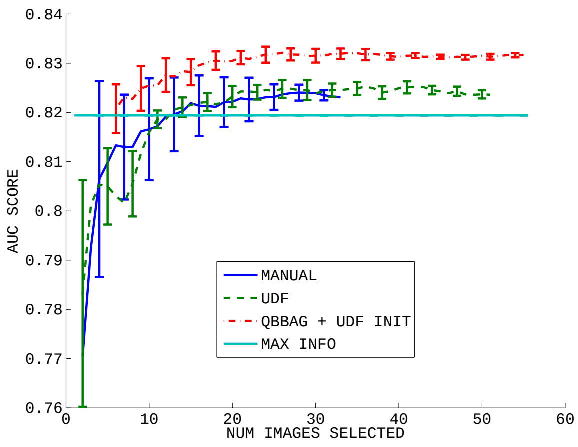
Figure 4.18 presents the results of similar experiments on the `FmObsCourseAll` data set. Compared to the previous data set, this one contains thin vertical poles, which make the laser data important. As a result, the plots for all three feature sets are indicating that active learning methods achieve good performance sooner and with higher likelihood than RANDOM selection. For this data set, active learning seems to also slightly outperform MANUAL selection (right column), as the confidence intervals for the AUC plots are much narrower. Interestingly, for the Color feature set, MANUAL and active learning methods outperform the MAX INFO classifier. This is an indication that the data distributions of the training and test sets were different, and that by ignoring some of the available data in the training set the MANUAL and active learning methods achieved better performance on the test set.

Figures 4.19 and 4.20 present results on two data sets on which, in reality, it makes little sense to apply active learning techniques: `FmDriveDown` and `FmVariousObs`. What makes these data sets different is their high density of image containing obstacles. As we have mentioned in Section 2 of this chapter, most of the images in the `FmDriveDown` data set contain fences, poles on the side of a road and barns, while the `FmVariousObs` was collected specifically to have a very high density of obstacle scenes. As a result, we expect random selection to perform extremely well on these data sets.

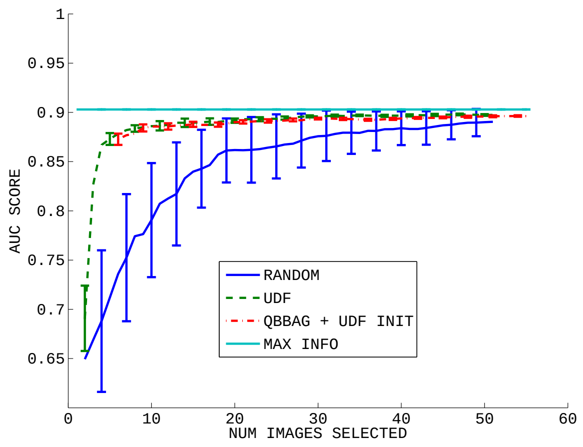
The results confirm our expectation, showing a significant overlap of all the different methods (MANUAL, RANDOM and active learning) on these two data sets. In general, the confidence intervals for the active learning algorithms are slightly narrower, which is good considering the comparable AUC score averages.



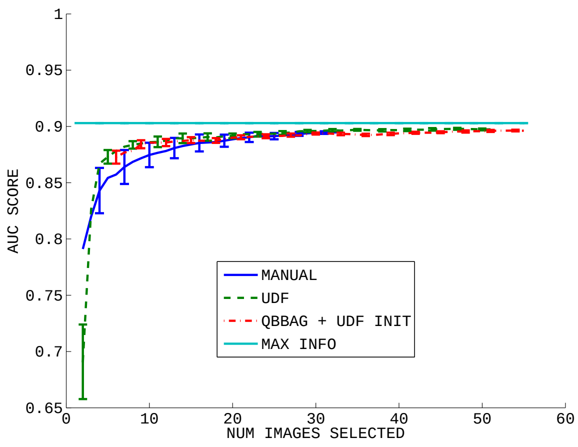
COLOR (AL vs. Random)



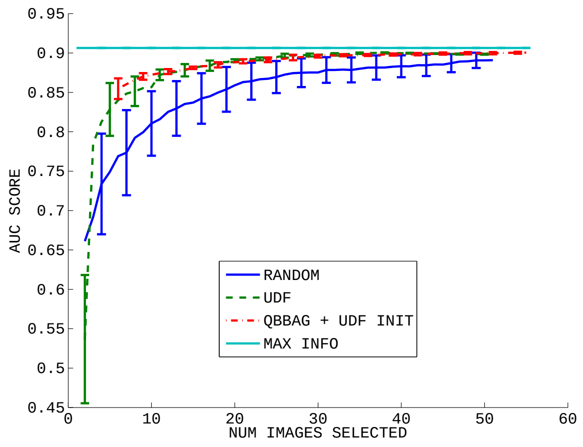
COLOR (AL vs. Manual)



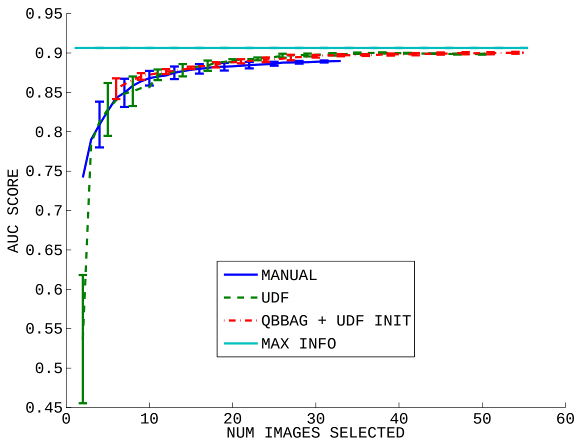
COLOR + TEXTURE (AL vs. Random)



COLOR + TEXTURE (AL vs. Manual)



COL + TEX + LASER (AL vs. Random)



COL + TEX + LASER (AL vs. Manual)

FIGURE 4.18. Results on the FmObsCourseAll data set.

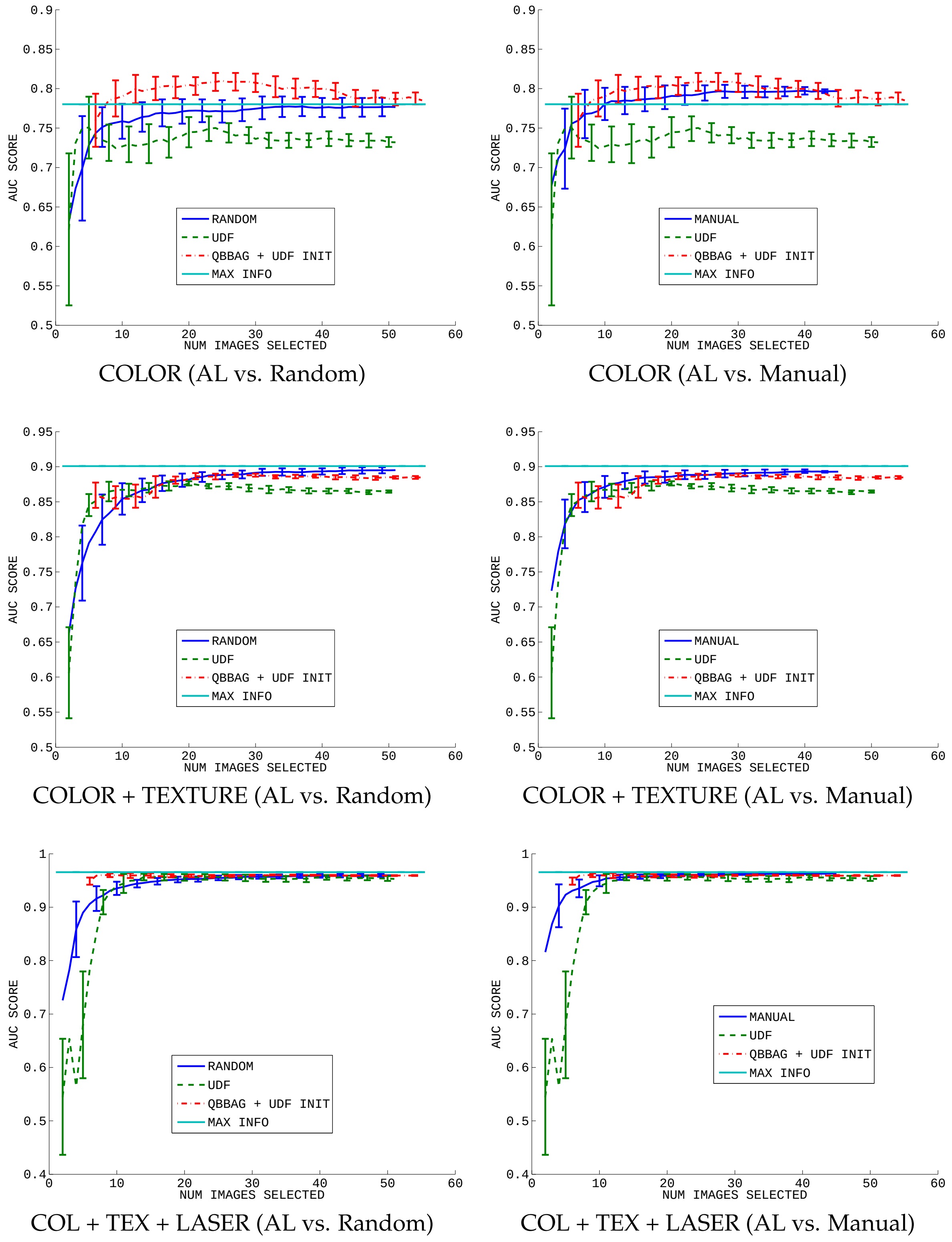
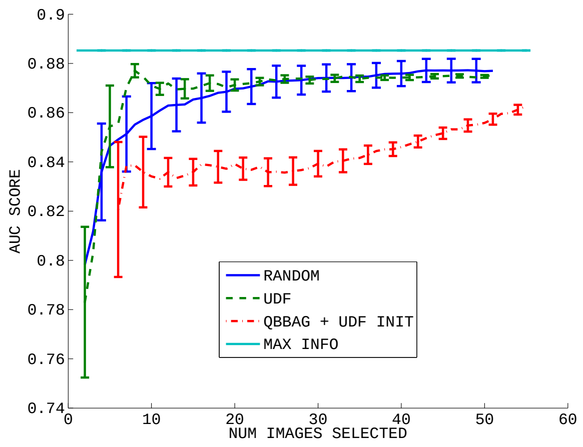
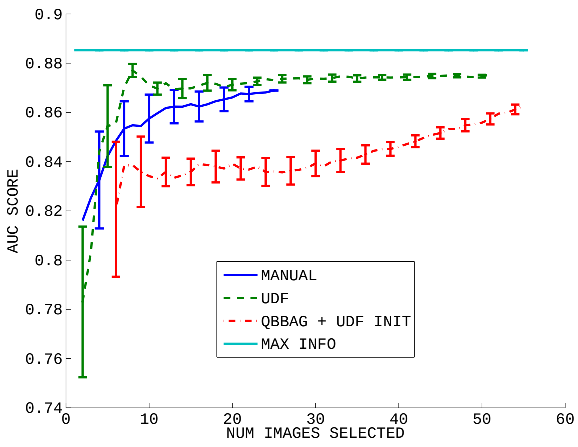


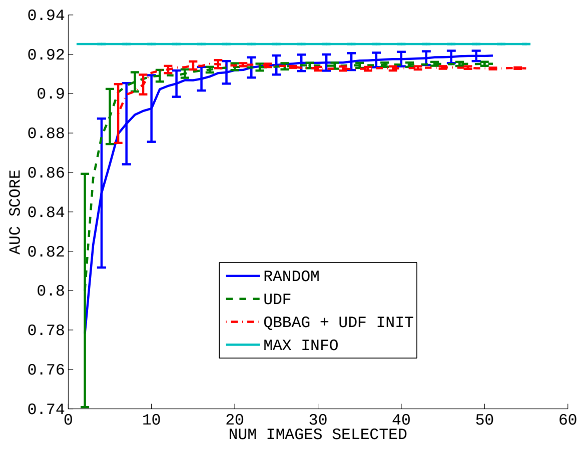
FIGURE 4.19. Results on the FmDriveDown data set.



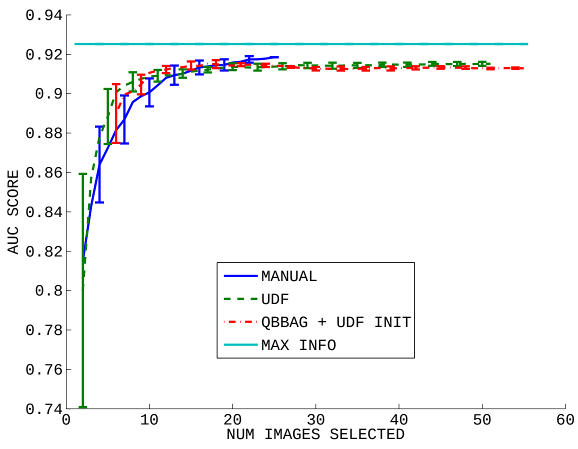
COLOR (AL vs. Random)



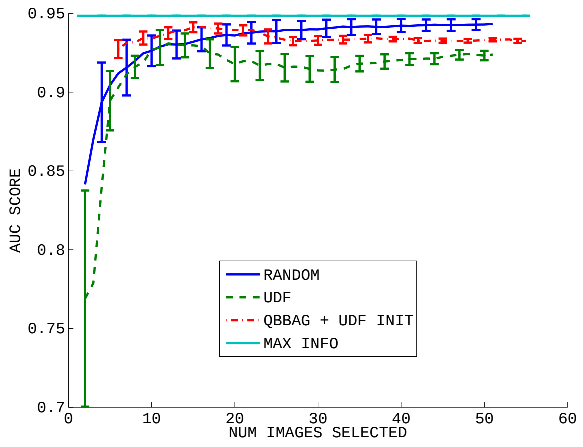
COLOR (AL vs. Manual)



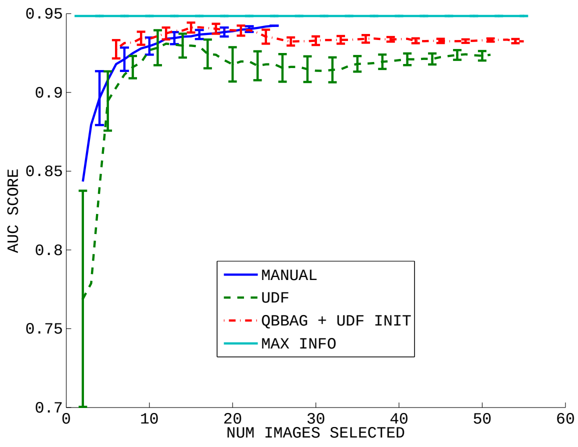
COLOR + TEXTURE (AL vs. Random)



COLOR + TEXTURE (AL vs. Manual)



COL + TEX + LASER (AL vs. Random)



COL + TEX + LASER (AL vs. Manual)

FIGURE 4.20. Results on the FmVariousObs data set.

The almost identical performances achieved by MANUAL and RANDOM selection indicate that we should not expect improvements from active learning. Nevertheless, it is important to confirm that active learning does not fail on these type of data sets.

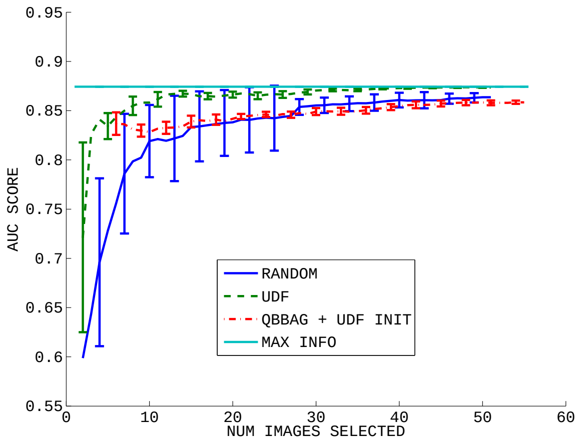
The results on the `ApMdSpiral` data set, the one we consider the most representative for our problem of interest due to its size and content, are presented in Figure 4.21. The results are very good for all three sets of features, showing that both UDF and QBBAG achieve better performance faster than RANDOM selection, and with much tighter confidence bounds. The performance is also comparable to MANUAL selection, although QBBAG is doing slightly worse on the Color feature set.

We believe these results prove that the active learning techniques we presented in this thesis can significantly reduce the amount of data labeling required in order to achieve good generalization performance. The data sets presented results on are challenging and very realistic, which gives us confidence that active learning is indeed relevant to real-world problems in our domain. Although the data sets we considered for this quantitative experiments were relatively small (with at most 2000 images), we will present results that show that they scale to much larger data sets.

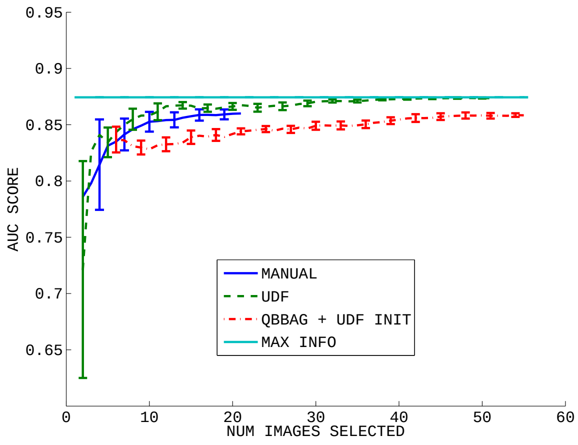
5.2. Comparing UDF, Manual and Random Initialization

One of the contributions of this thesis is the introduction of the Unlabeled Data Filtering algorithm, which is our solution for the initialization problem affecting many standard active learning algorithms. Although we have shown in the previous section that UDF works remarkably well as a stand-alone active learning method, we have investigated its performance as an initialization method and we present the results in Figures 4.22 through 4.26.

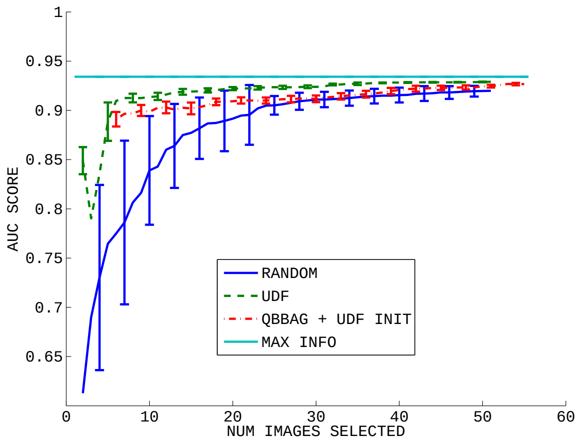
Not surprisingly, using Unlabeled Data Filtering for initialization results in slightly improved performance over RANDOM initialization on the data sets with moderate class imbalance (Figures 4.22 and 4.23), in no improvement for the data



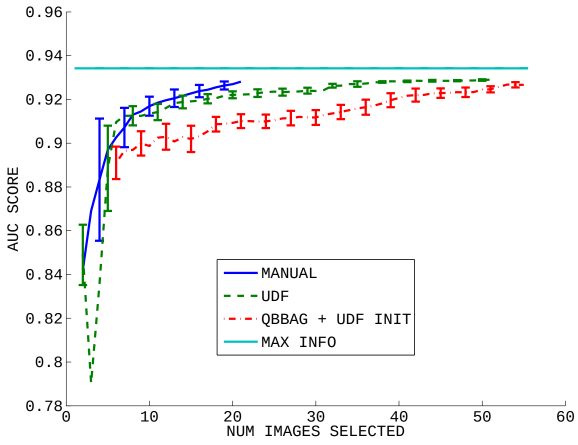
COLOR (AL vs. Random)



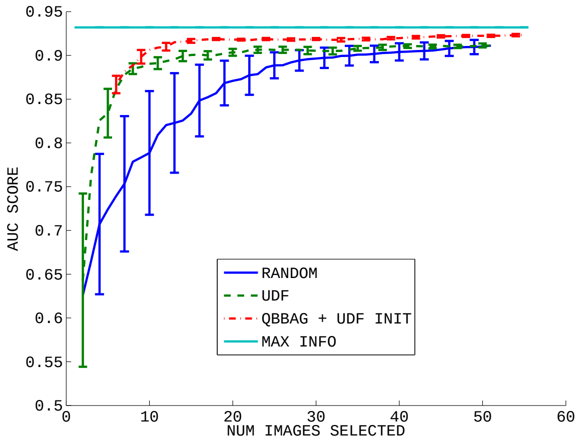
COLOR (AL vs. Manual)



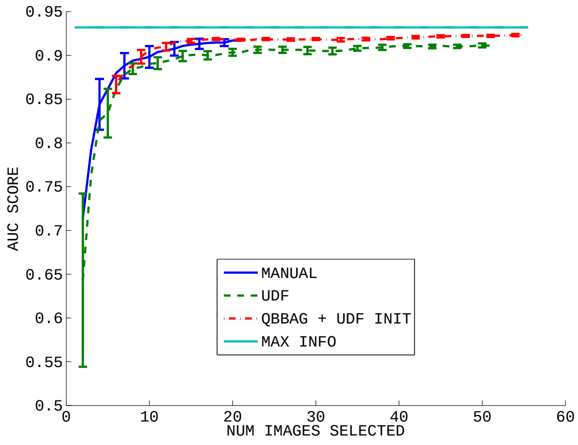
COLOR + TEXTURE (AL vs. Random)



COLOR + TEXTURE (AL vs. Manual)

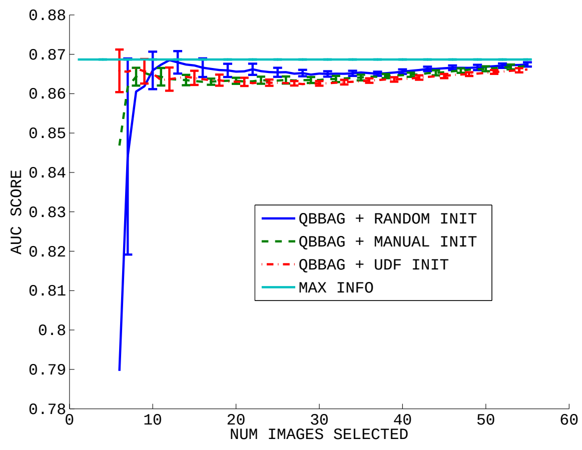


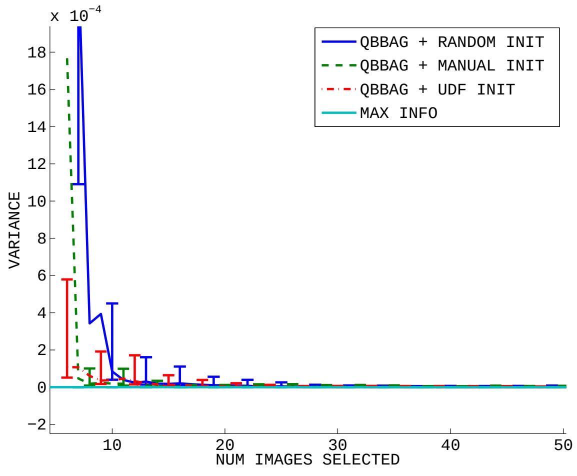
COL + TEX + LASER (AL vs. Random)



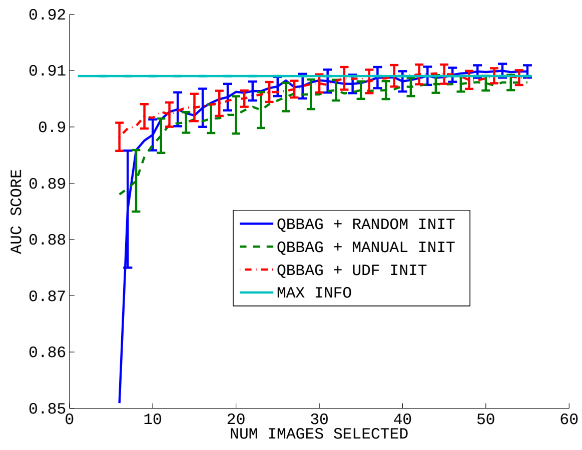
COL + TEX + LASER (AL vs. Manual)

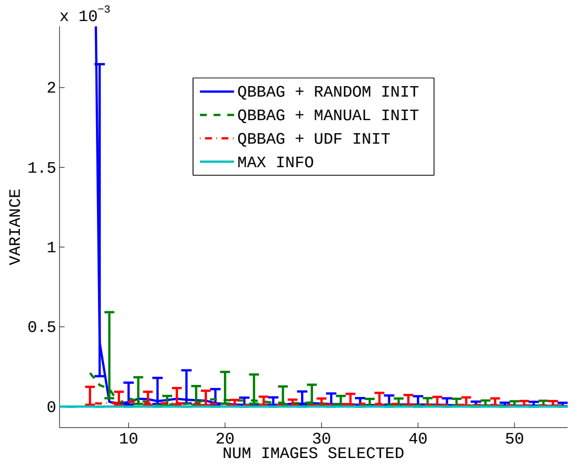
FIGURE 4.21. Results on the ApMdSpiral data set.



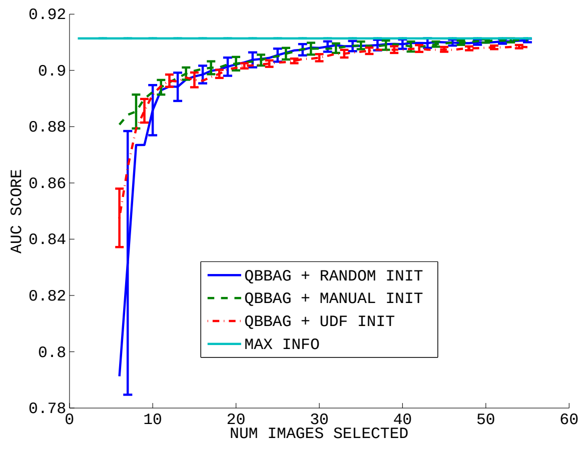


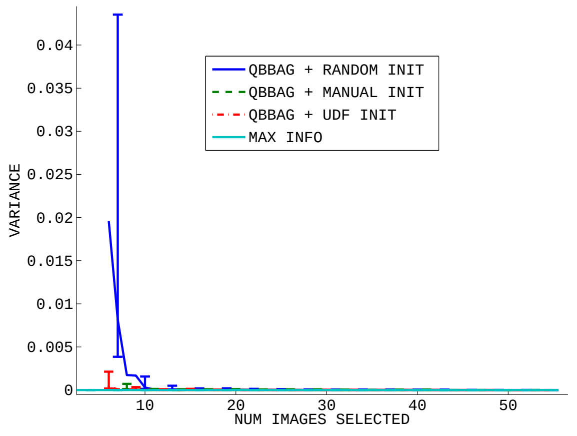
COLOR





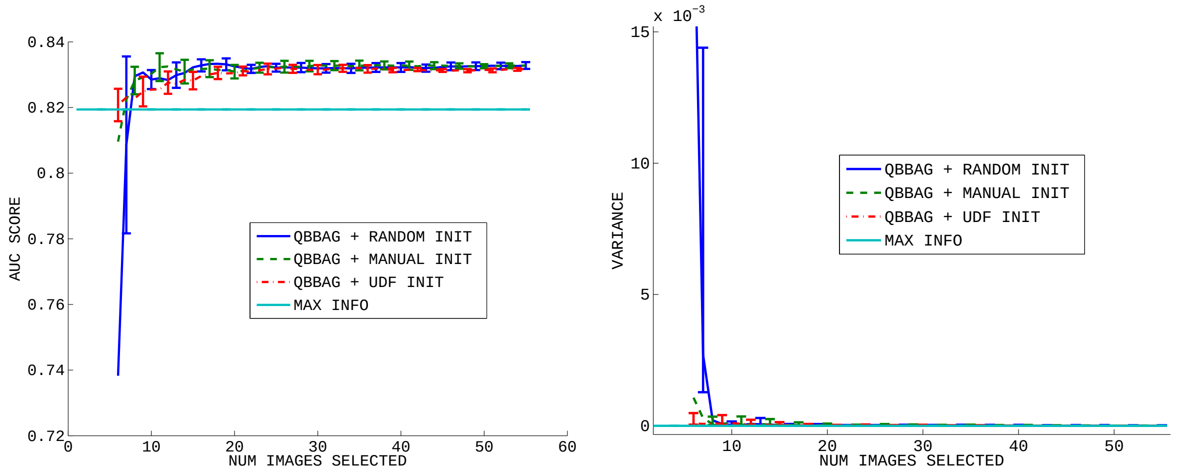
COLOR + TEXTURE



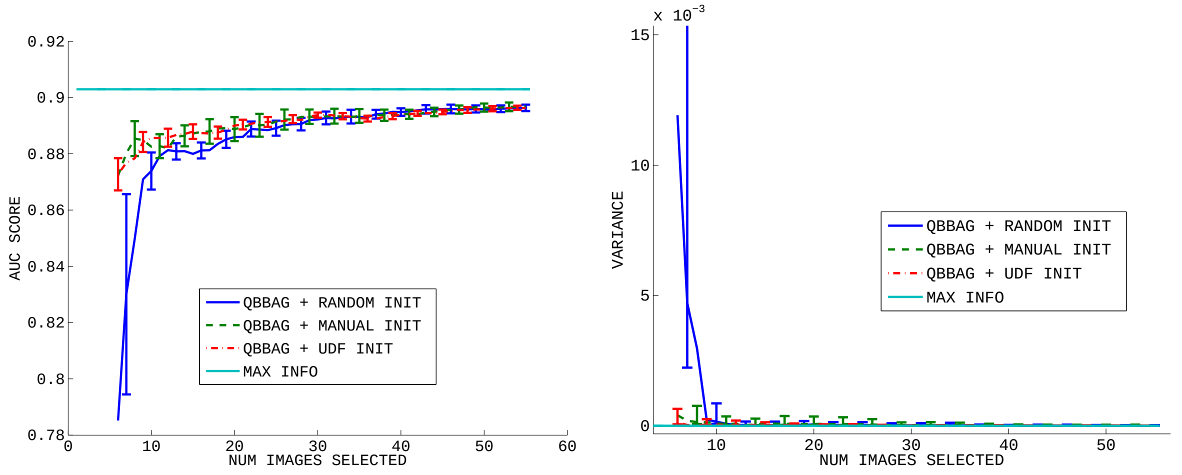


COL + TEX + LASER

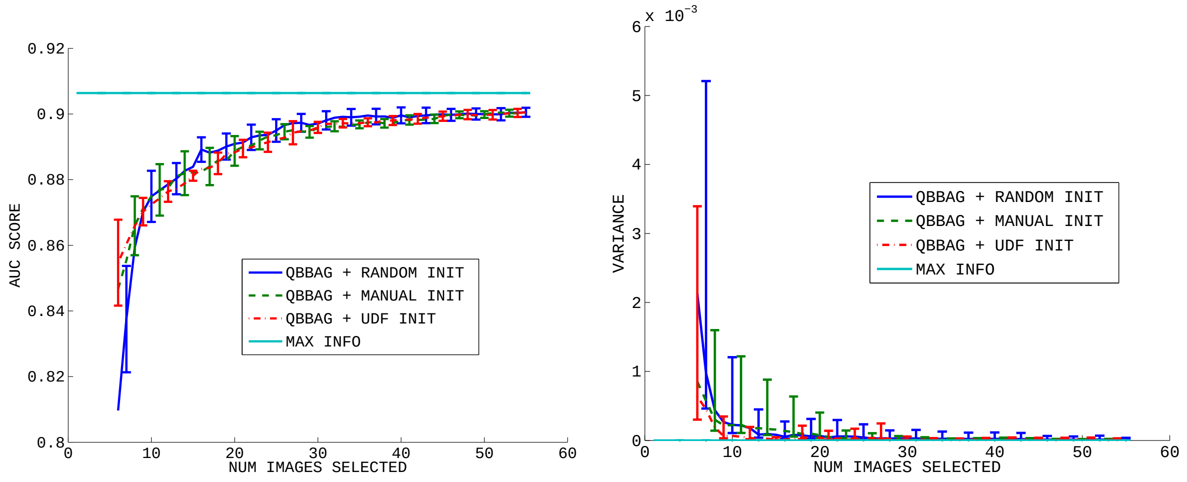
FIGURE 4.22. Comparing RANDOM, MANUAL and UDF initialization for the QBBAG algorithm on the FmObsCourseNonPoles data set.



COLOR

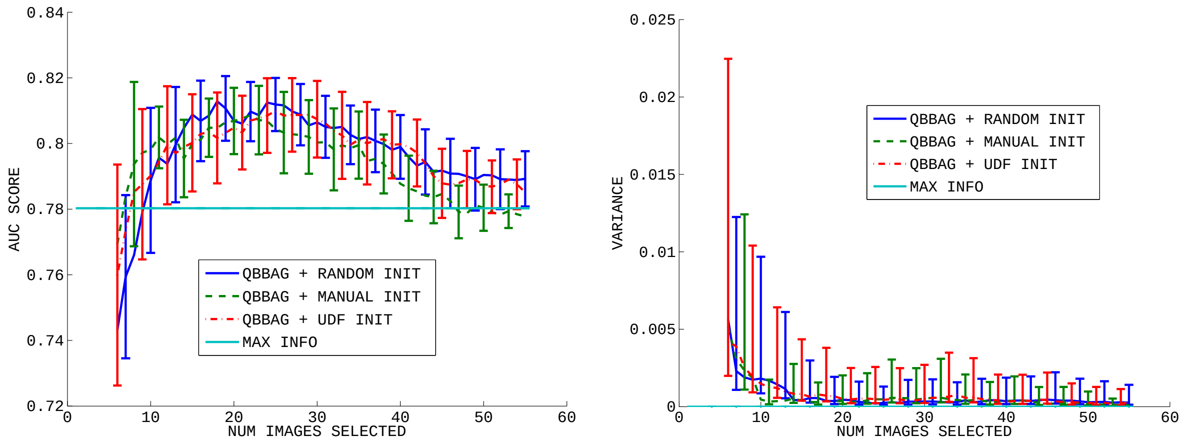


COLOR + TEXTURE

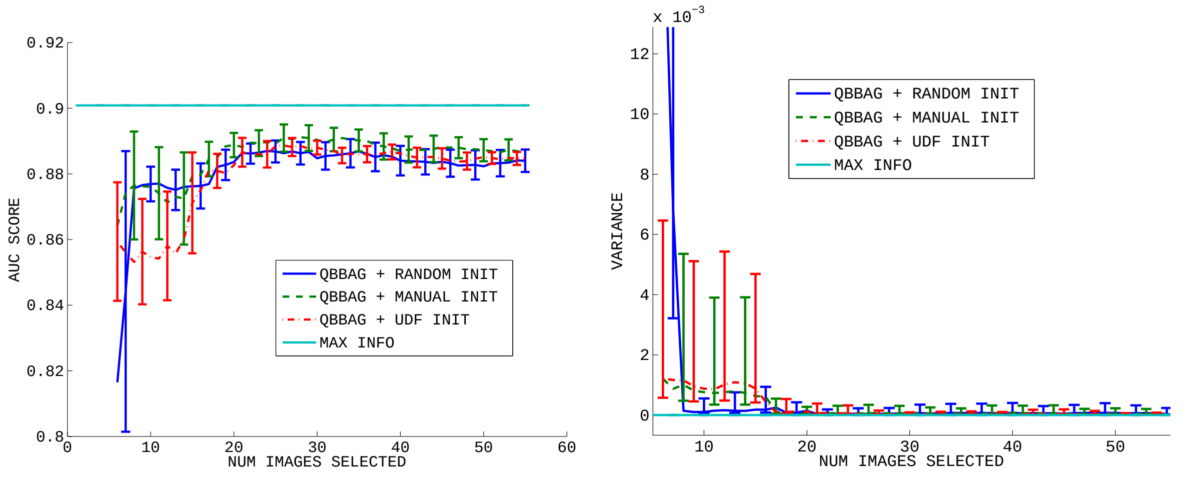


COL + TEX + LASER

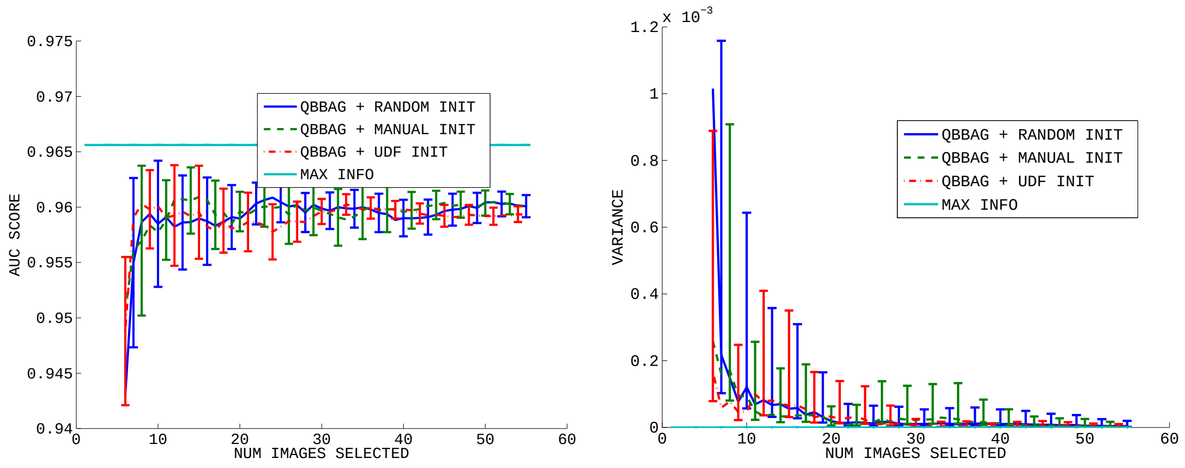
FIGURE 4.23. Comparing RANDOM, MANUAL and UDF initialization for the QBBAG algorithm on the FmObsCourseAll data set.



COLOR

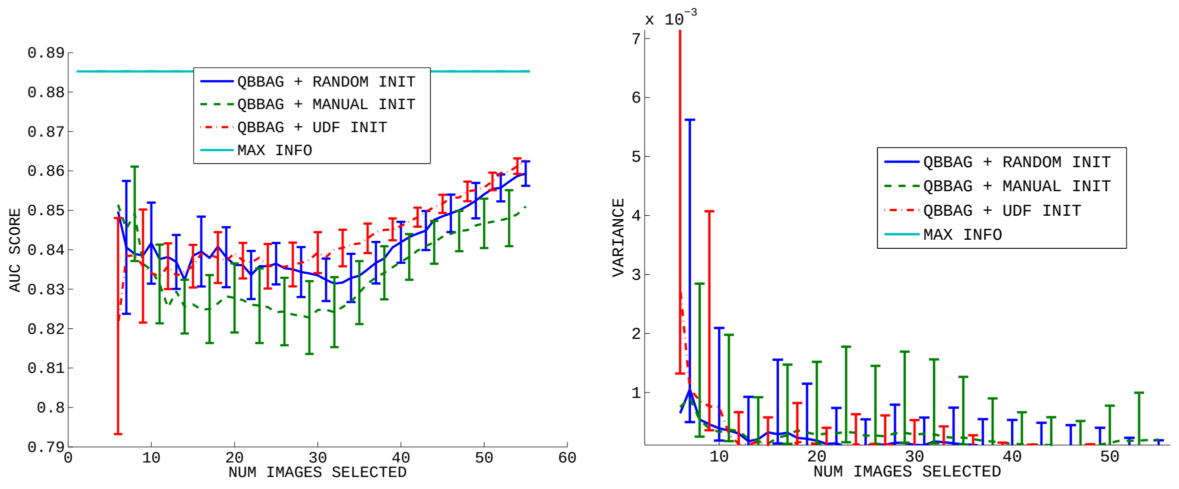


COLOR + TEXTURE

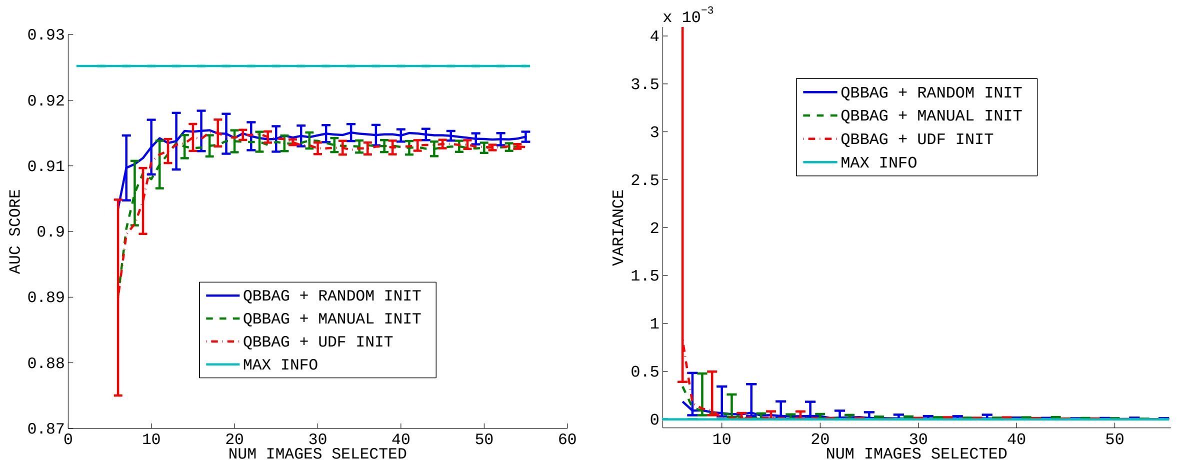


COL + TEX + LASER

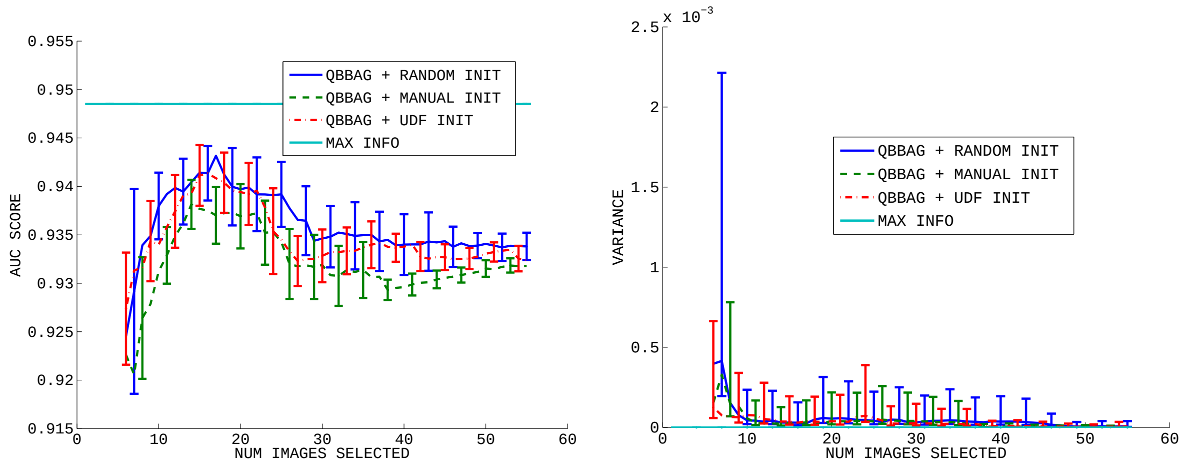
FIGURE 4.24. Comparing RANDOM, MANUAL and UDF initialization for the QBBAG algorithm on the FmDriveDown data set.



COLOR



COLOR + TEXTURE



COL + TEX + LASER

FIGURE 4.25. Comparing RANDOM, MANUAL and UDF initialization for the QBBAG algorithm on the FmVariousObs data set.

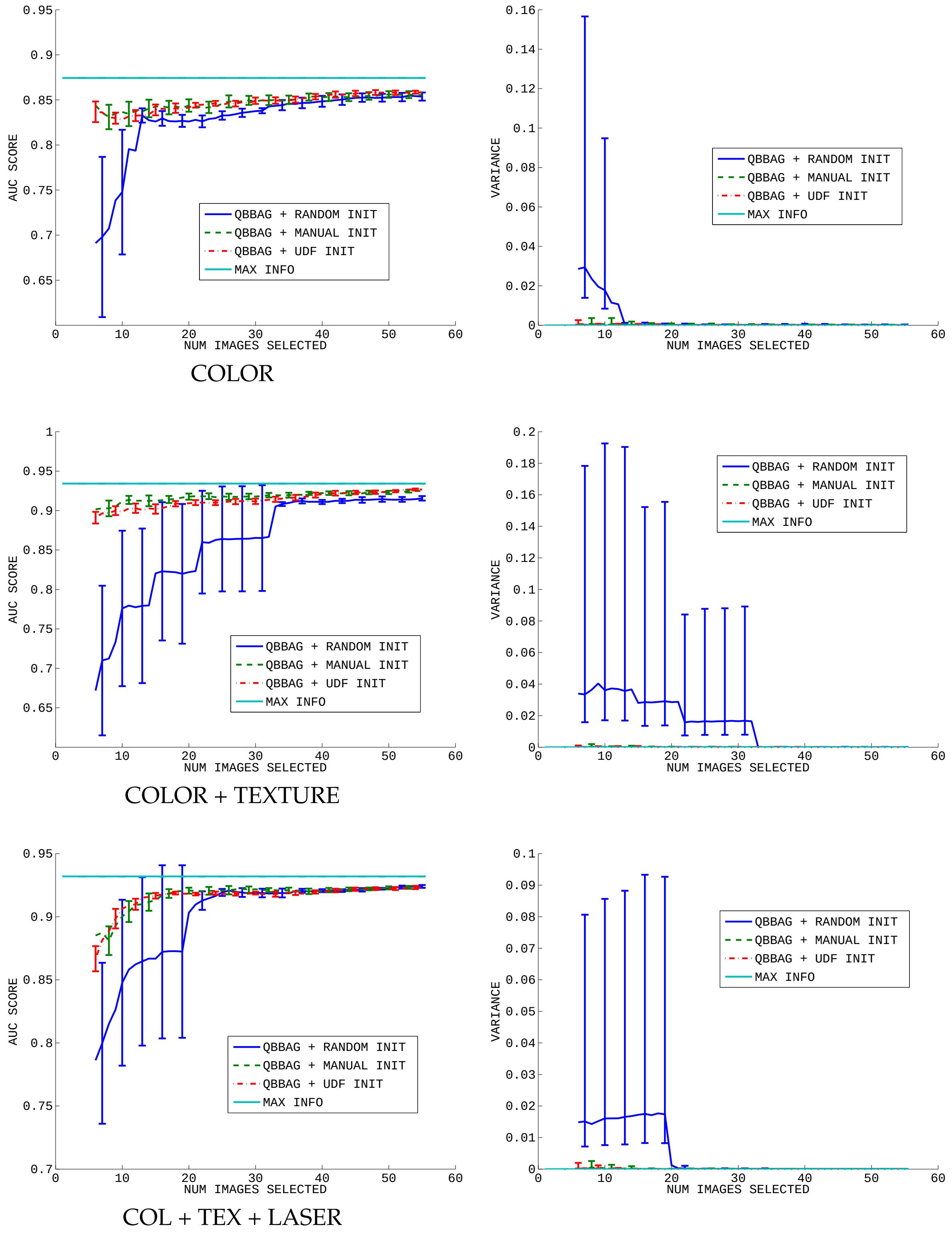


FIGURE 4.26. Comparing RANDOM, MANUAL and UDF initialization for the QBBAG algorithm on the ApMdSpiral data set.

sets with no class imbalance (Figures 4.24 and 4.25) and in a dramatic improvement on the `ApMdSpiral` data set that presents severe class imbalance (Figure 4.26).

On the `FmObsCourseNonPoles` data set, UDF initialization leads to a much smaller confidence interval for the starting point of the active learning algorithm, for all three sets of features in Figure 4.22. The same is true for the results obtained on the `FmObsCourseAll` data set (Figure 4.23), except that the gains are slightly smaller for the (Color + Texture + Laser) feature set. To better illustrate the evolution of the confidence bounds for the mean AUC score as more images are labeled, on the right columns of Figures 4.22 through 4.26 we display separate plots of the standard deviations of the AUC score for each algorithm, for each number of images selected.

On the two data sets with no class imbalance, `FmDriveDown` and `FmVariousObs`, the improvements of the QBBAG performance when using UDF initialization either small (Figure 4.24 for (Color + Texture) and (Color + Texture + Laser) features) or inexistent (Figure 4.24 for Color features, or Figure 4.25). While this is not a great result in itself, given the nature of these data sets we did not expect to see significant gains when using UDF for initialization: for data sets where the "interesting" patterns are very frequent, random selection for initialization is a very good solution.

As it was the case for the active learning vs. random selection experiments, the results on the `ApMdSpiral` data set are excellent, since its unbalanced class priors perfectly match the conditions we designed the UDF algorithm for. The plots in Figure 4.26 show that the MANUAL and UDF initialization routines lead to excellent performance compared to RANDOM selection, both in terms of the average performance and the confidence intervals. This is not at all surprising given the small frequency of interesting patterns in this data set.

To show that the good performance obtained with UDF initialization is not related to the specific active learning algorithm used, in Figure 4.27 we present results obtained on the `ApMdSpiral` data set with the Co-Testing algorithm, using

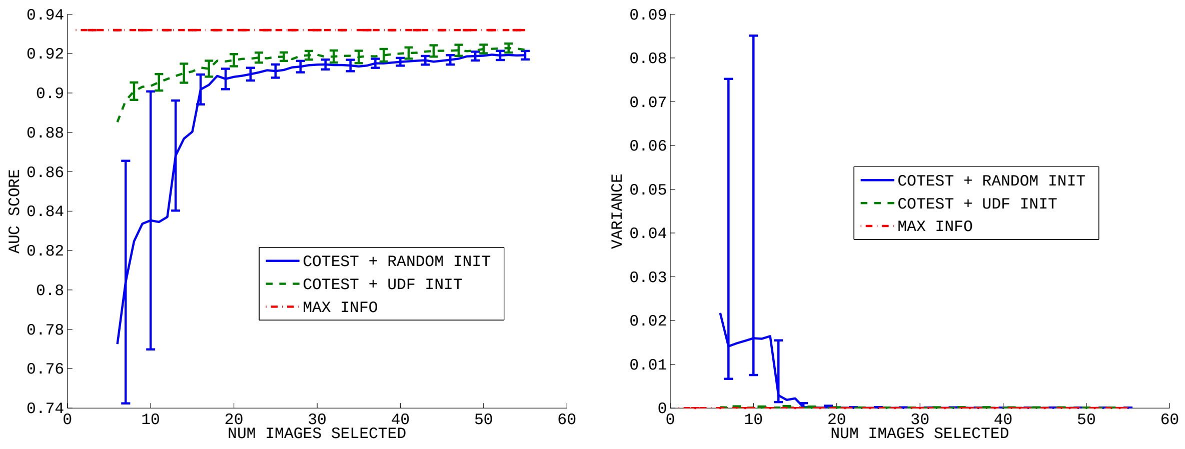


FIGURE 4.27. Comparing RANDOM and UDF initialization for the COTEST algorithm on the ApMdSpiral data set. We have used Color, Texture and Laser features as three views of the data. The disagreement between the views was measured using the KL-divergence-to-the-mean.

Color, Texture and Laser as its three views of the data. We used KL-divergence-to-the-mean to measure disagreement between the views. The results are very similar to the ones obtained with QBBAG, showing that using UDF for initialization leads to much faster learning than when initializing randomly.

To conclude this round of experiments, we draw the conclusion that the Unlabeled Data Filtering algorithm is indeed an effective technique for initializing standard active learning algorithms, especially in cases where significant class imbalance occurs. In such cases, its performance is similar to MANUAL initialization and much better than RANDOM initialization.

5.3. Imposing Spatial Constraints for Aggregating Interest Scores

One of the interesting aspects involved in adapting active learning to outdoor robotic perception was the need to address the data block constraint. As we have showed in Section 2 of Chapter 3, interest scores assigned to the individual image patches by our algorithms need to be aggregated into a total image score, allowing us to select interesting images for labeling.

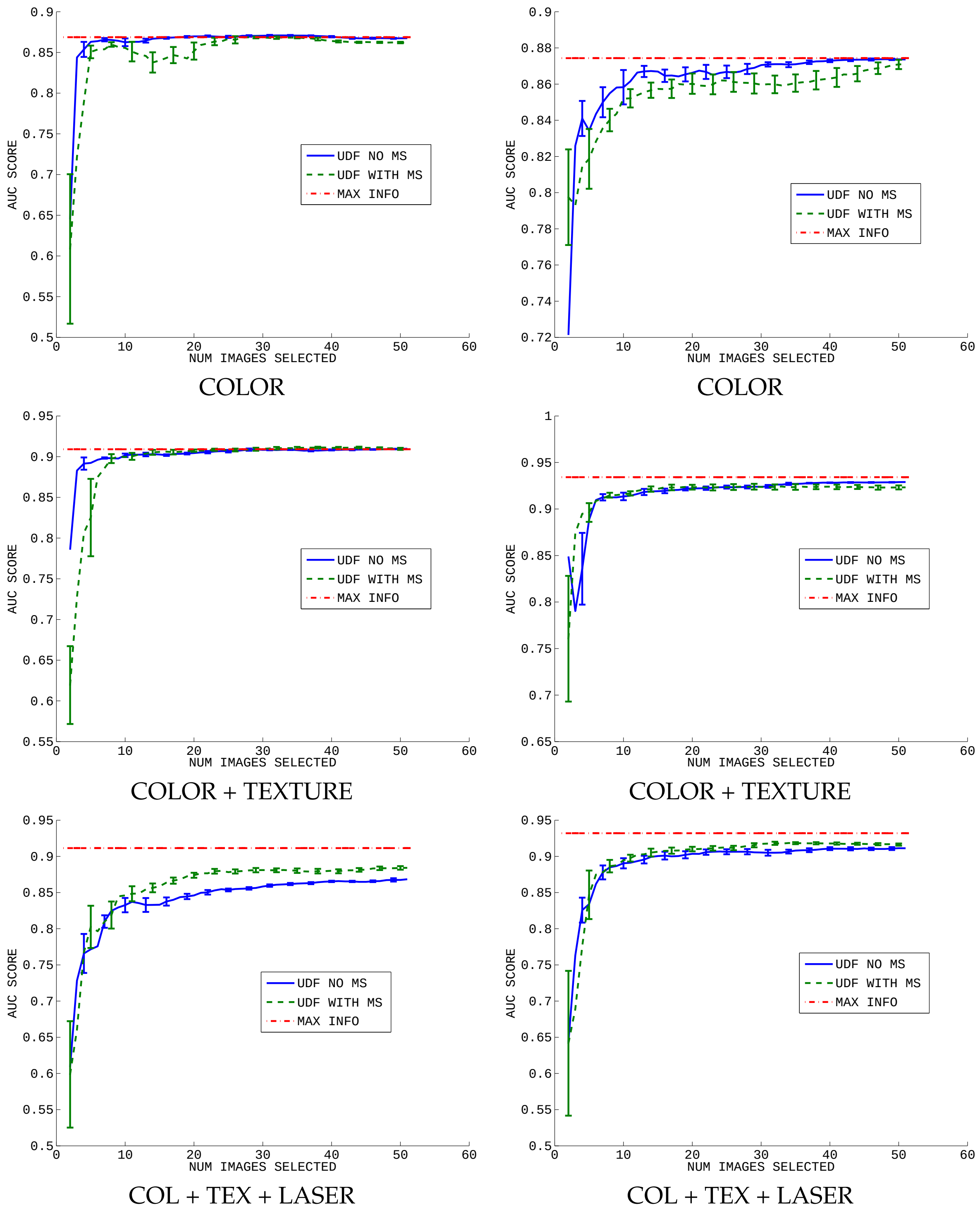


FIGURE 4.28. The effect on UDF of using Mean-Shift segmentation to impose spatial coherency constraint. The results presented are based on the FmObsCourseNonPoles (LEFT) and ApMdSpiral (RIGHT) data sets.

The first solution we proposed was to average the interest scores of the k most interesting image patches in each image[10]. The second solution involved segmenting the image score in coherent regions, and compute the interest score of each coherent region independently. Each image gets assigned the score of the highest scoring region. We used the Mean Shift algorithm [**12**] for segmenting the image score. In this section, we are comparing the results of the two aggregating schemes, presenting results with the UDF algorithm on the `FmObsCourseNonPoles` and `ApMdSpiral` data sets, using Color, (Color + Texture) and (Color + Texture + Laser) features.

Unfortunately, the results we obtained in these experiments are not very conclusive. Based on the results in Figure 4.28 and on other experiments we performed, it seems that imposing spatial coherency constraints is slightly harmful when using Color features, almost indifferent on (Color + Texture) features, and slightly beneficial on (Color + Texture + Laser) features. Our current belief is that the (Color + Texture + Laser) feature set leads to the noisiest score images because of the laser features, which means that the aggregation scheme without spatial constraints is more likely to combine the interest scores of disparate image patches. This is only a hypothesis at this point.

The main problem with our experiments came from our choice of using the mean shift algorithm for imposing spatial coherency. After a more in-depth analysis of our results, we have discovered that generally, the segmentations produced on our image scores were quite poor, often disagreeing with the clusters we could easily identify visually in the data. For specific data sets and feature sets, we were able to tune the parameters of mean shift so that its results agreed with the desired segmentation; however, these parameter settings were rarely leading to good segmentations of new data sets. Since we have chosen to perform all our experiments without changing parameters in between, the results we present in Figure 4.28 were obtained based on sub-optimal segmentation parameters.

[10]We used $k = 8$ for the experimental results presented, as indicated in Section 4.1

Although these experiments were quite inconclusive, we continue to believe that taking into account the spatial distribution of the image patches used for computing the image interest score is a good idea, that could lead to accelerated learning. The small improvement we have observed on the noisy (Color + Texture + Laser) feature set –despite our unfortunate choice of the method for imposing the spatial constraints–, gives us reasons to hope that future work in this area can lead to more substantial improvements. Until then, the first aggregation solution we proposed can be used reliably for many practical applications in our domain.

5.4. Scaling Up

The primary motivation for our research in active learning techniques for outdoor perception was to enable the use of supervised learning on large scale, real-world problems. The experiments we have presented so far demonstrate that active learning can significantly reduce data labeling requirements, which is a necessary condition for our goal. An additional requirement is that the methods have scaling properties which make them feasible for real-world applications.

When gathering the experimental evidence we presented so far, we have focused on obtaining precise quantitative results, that allow us to make direct comparisons and obtain significance estimates. The requirement that a large number of randomized experiments be easily performed has in turn constrained us to only consider data sets that could be exhaustively labeled. For this purpose, we have labeled approximately 8,000 images. The largest data set we had available contains 2,000 images and was recorded during a one mile-long trip.

While obtaining results on a 2000 image data set with millions of image patches is not negligible, it is certainly far from convincing that our methods would scale up to the much larger data sets that would be encountered in true real-world applications. Instead of looking at one mile of travel, we are interested in applying our methods to tens or hundreds of miles.

As a first step in that direction, we have performed preliminary qualitative experiments on a data set of approximately 50,000 images. The data was recorded

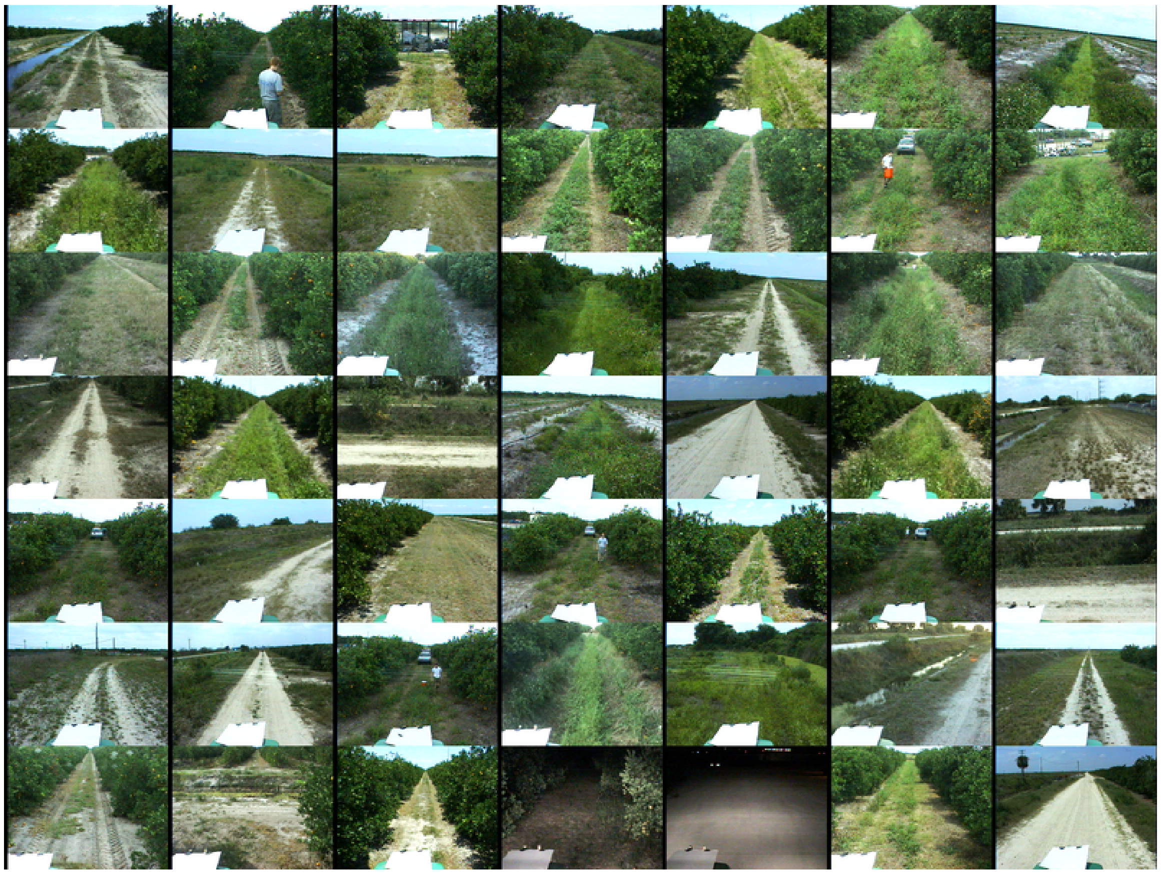


FIGURE 4.29. Images selected randomly from the 50,000 image data set.

during several days of data logging in an orange grove in Florida, in 2000. As a result, the amount of variation of the patterns contained in this data set is huge: there are night scenes, man-made structures, people, irrigation canals, different types of vegetation, obstacles, cars, etc. No images were removed from the data set, other than a short sequence where the camera malfunctioned, generating unusable images.

Figure 4.29 contains an image mosaic obtained by randomly selecting 50 images. Similarly to the mosaic generated by random selection on the `ApMdSpiral` data set (see Figure 4.12), a large percentage of the images selected consist of relatively non-interesting vegetation. There are two images from the night scene (bottom row) and a few images of a person in a distance. Notice however that RANDOM selection had a good chance of selecting the human images, since we left the cameras recording data as we spent 15-20 minutes setting up an obstacle course in

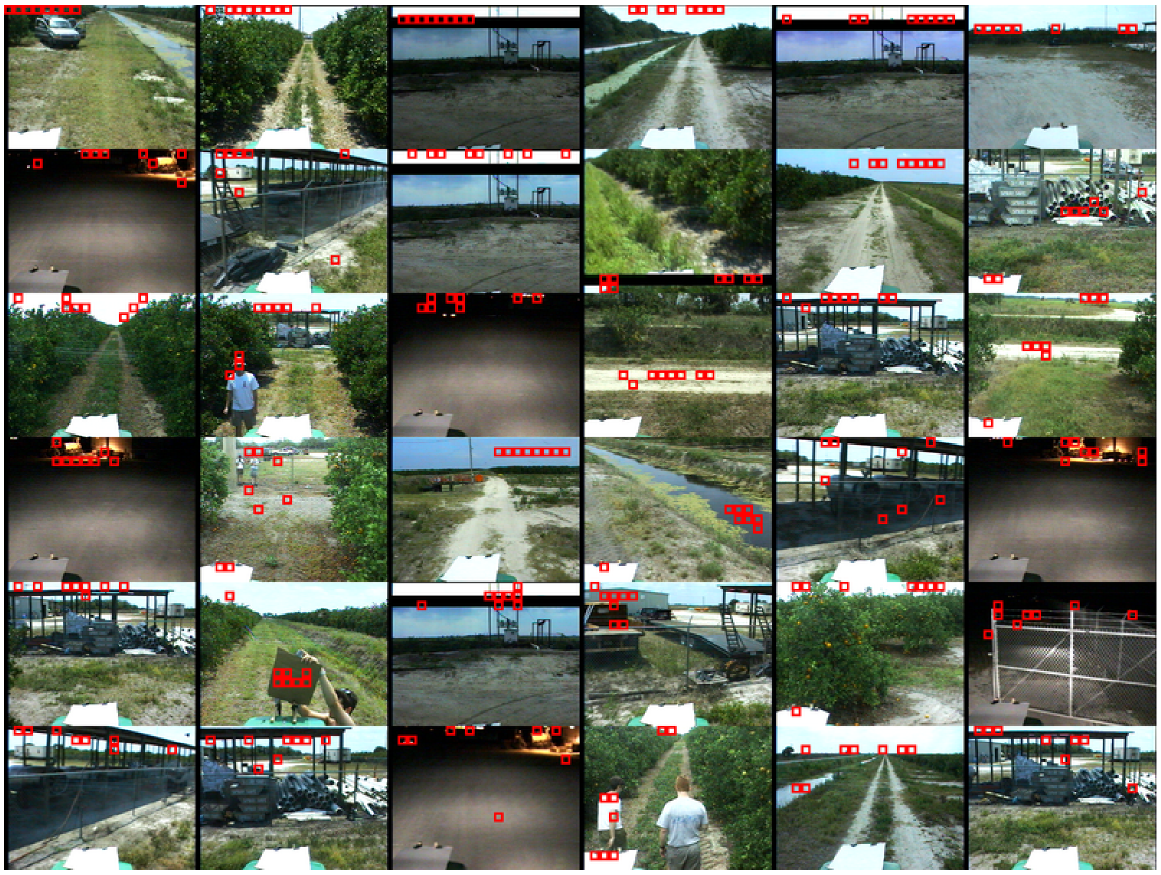


FIGURE 4.30. Images selected by the UDF algorithm from the 50,000 image data set.

front of our vehicle. Overall, we consider that the subset of images selected by the RANDOM method represents only a very small part of the variation contained in the original data set.

We contrast the mosaic obtained by random selection with the one presented in Figure 4.30, which was obtained using the Unlabeled Data Filtering algorithm on color and texture features. The algorithm, designed to find "interesting" patterns, revealed information we found indeed interesting: the data set contained another sequence recorded with a malfunctioning camera (other than the one already removed), which resulted in a small black stripe at the top of the image. We found this result very positive, considering that we were unaware of this significant problem with the data set. For the experiments described bellow, we have chosen not to label these corrupted images and not to replace them with other images selected by UDF.

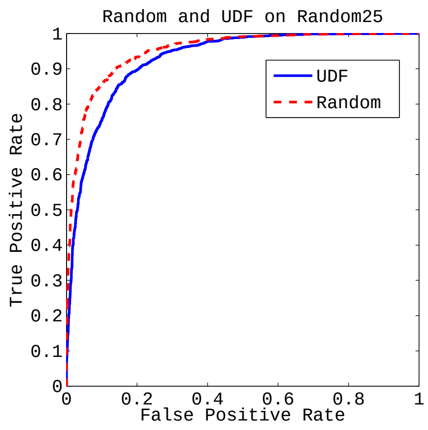
Other than the corrupted images, UDF selected a large number of interesting scenes: close-up views of people, a calibration target, many man-made structures, a night shot of the road seen under headlight illumination, some interesting effects of artificial illumination at night and water in an irrigation canal. For this experiment we did not prevent the algorithm from considering sky patches, and a number of images were selected because of interesting sky colors. In a real application, such examples can generally be discarded.

Based on a qualitative analysis of these results, one would expect that a classifier trained using the UDF selected images for detecting obstacles or artificial structures would perform better than a classifier trained using the randomly selected images from Figure 4.29.

To verify this intuition, we have generated four labeled data sets from the 50,000 image set by segmenting the images in traversable and non-traversable terrain. The images in each data set were selected as follows:

- We labeled the 50 randomly selected images presented in Figure 4.29, and combined them in the TrainRANDOM data set.
- We labeled the 44 non-corrupted images selected by the UDF algorithm and combined them in the TrainUDF data set.
- We selected 25 more images randomly and we labeled and combined them in the TestRANDOM data set.
- We analyzed the entire set of images and manually selected and labeled 40 images that contain all the major types of obstacles that we encountered. The traversable areas from these 40 images were labeled as well. This data set is referred to as the TestOBSTACLE data set.

We have trained two logistic regression classifiers using color and texture features on the two training data sets, TrainRANDOM and TrainUDF. The ROC curves



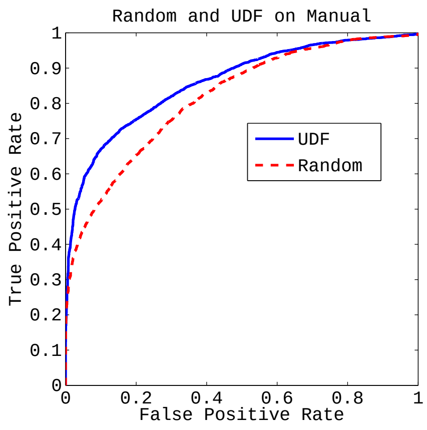


FIGURE 4.31. ROC curves obtained by the classifiers trained on the TrainRANDOM and TrainUDF data sets to the TestRANDOM (left) and TestOBSTACLE (right) data sets.

obtained when applying these two classifiers to the TestRANDOM and TestOBSTACLE data sets are presented in Figure 4.31. The results indicate that as we hoped, the classifier trained using the UDF selected data can better classify the important patterns in the data, which correspond in our application to obstacles. For this experiment, this comes at the cost of a slightly reduced performance on the randomly selected test set.

The fact that even the classifier trained on UDF selected data has relatively low performance on the manually selected data set –although its training set was representative for the classification problem– indicates that our feature set is not sufficiently powerful for this application. This is not surprising, given that the input images had low resolution (320x240) which limits the use of our texture features, and that color is a weak feature for this environment containing many shades of green grass (non-obstacle) and green trees (obstacles).

To verify this hypothesis, we have generated a few classification images using the two classifiers. The results are presented in Figure 4.32. We notice that the classifier trained on the randomly selected images fails to detect the ladder and the gate, and produces a much weaker response for the human obstacle. The classifier trained on UDF selected data correctly classifies those obstacles, but it produces higher outputs for many instances of the non-obstacle class (as one would expect,

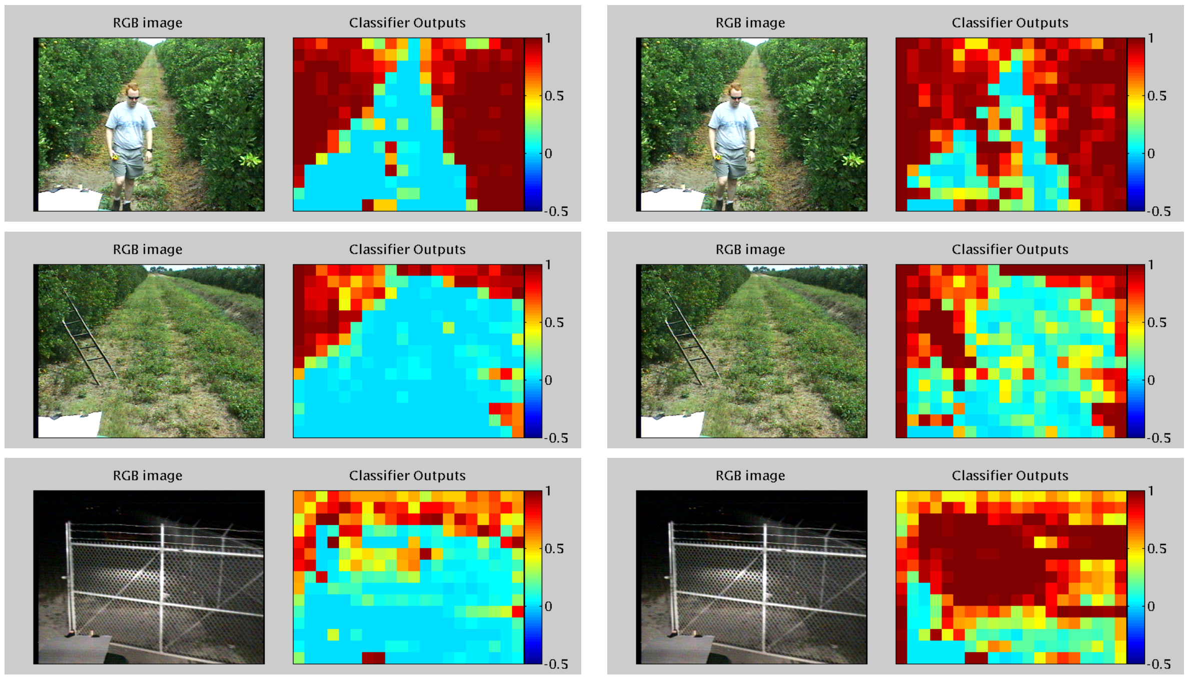


FIGURE 4.32. Classification examples obtained by applying the classifiers trained on the TrainRANDOM (left) and TrainUDF (right) data sets to scenes from the test set. The classifier trained on randomly selected data fails to detect the gate and the ladder and only produces a weak return for the person. The classifier trained on UDF selected data detects those obstacle but produces slightly higher responses for the non-obstacle class.

looking the the ROC curve it produced for the TestRANDOM data set). The fact that the UDF trained classifier has difficulty discriminating between grass and trees although it was exposed to sufficient training examples of both indicates that indeed the features are too weak to allow logistic regression to separate both the trees and the other obstacles from the grass. These results are consistent with our intuition that using UDF for training is somewhat equivalent to a reweighting of the training examples that favors rare examples from the less frequent class. In certain situations where the classifier has insufficient capacity, this can be a drawback.

In conclusion to our experiments on very large data sets, we believe that additional work is necessary before the algorithms we used to generate these results are ready for real use on data sets of tens of thousands of images: we would need to reduce the computational requirements, and try to improve the efficiency of our methods even further. Nevertheless, the fact that such large data sets are still within the realm of possibilities for a Matlab implementation of our algorithms gives us hope that their scaling properties make them fit for the large learning problems that need to be solved in the outdoor perception domain.

6. Conclusions

To conclude the chapter on our experimental evaluation, we will summarize our main findings:

- Active learning can significantly reduce the amount of data labeling required for applying supervised learning techniques to our domain, especially on problems presenting unbalanced class priors. On such data sets, both UDF and QBBAG (initialized with UDF) outperform random selection, and have similar effectiveness to manual selection by a human expert (Section 5.1, page 81).
- Selecting for labeling unusual data patterns using the Unlabeled Data Filtering algorithm is an effective method for initializing active learning algorithms. The benefits of initializing using UDF compared to random selection become more important as the unbalance between class priors grows (Section 5.2, page 92).
- The data block constraint that is particular to our domain of interest, can be addressed with or without taking into account the spatial coherency of the image patches determining the interest score of an image (a data block). Our experiments show that the method of simply averaging the interest score over the highest scoring image patches in an image is robust, and produces good results. Enforcing coherency constraints has the potential of improving performance even further, but the method we considered for doing so is not sufficiently robust. Further work is required in this area.
- The solutions we propose have good scaling properties, and are applicable to real-world sized problems (Section 5.4, page 103).

CHAPTER 5

Discussion and Future Work

IN the last few chapters, we have proposed a set of solutions to the data labeling problem. We believe our experimental results are a strong indication that the solutions are practical, and represent an important step towards making supervised learning techniques part of a standard "toolbox" for solving outdoor perception problems in our domain.

Aside from the fact that less data needs to be labeled and learning from very large data sets is possible, automated methods for selecting data for labeling make classifier training accessible to users that have no prior knowledge of machine learning, statistics or computer vision. The end user can train a perception system without paying attention to the sets of features or the specific learning algorithm used. We pride ourselves that after its initial development, the UDF algorithm has been packaged as a standard component of a system for terrain classification and obstacle detection. This system has been used for more than a year both internally within the Robotics Institute at CMU and externally by engineers evaluating the applicability of our classification system for various outdoor perception problems. The fact that the UDF algorithm already enabled users without any machine learning background to produce effective classifiers suggests that indeed, effective active learning algorithms will unlock an entire new set of applications of machine learning in our domain.

We consider that the main contribution of this thesis is not a specific method of applying active learning in our domain, but rather the signal that these techniques have a great role to play in outdoor robotics. New questions arised from our research effort, and many of them had to be relegated to future work.

A surprising finding of our experiments is the success of the Unlabeled Data Filtering algorithm as a stand-alone active learning method. The fact that an algorithm that is simply trying to uniformly cover the feature space can perform comparably to the best active learning method we have tested is somewhat unexpected. We would like to investigate in more detail the conditions under which being more subtle and taking into account estimates such as the expected version space reduction starts to pay off. A idea inspired to us by the work presented by Pelleg and More in [**64**] is to determine if by alternating between the UDF and QB-BAG data selection methods, one could obtain a meta-algorithm that has a better worst-case behavior.

A logical extension to the work we presented is to further explore solutions for imposing more constraints on what an interesting pattern is. As Wong et al. have suggested in [**91**] in the context of anomaly detection, we often risk detecting isolated outliers instead of interesting *patterns in the data*. Our preliminary effort in imposing spatial coherency constraints when accumulating interest scores was a step in the right direction, but more research is required on that topic.

A very interesting research direction would be to try to get a better theoretical understanding of why active learning works at all. Given that its main effect is that it alters dramatically the distribution of the training data, it is unclear if the resulting performance gains are the result of the implicit reweighting of the training data, or of the different spatial distribution of the training examples.

Last but not least, we hope that this thesis could help promote the learning approach for large scale perception problems, which will lead to an amount of experimentation vastly superior to what we could achieve in our effort. We have the certitude that this additional experimentation will lead the researchers in our field to great new problems, which will help push forward the state of the art in outdoor mobile robotics.

APPENDIX A

UDF: A Few Implementation Details

In our presentation of the Unlabeled Data Filtering algorithm we have tried to make it possible for other researchers to implement their own version of the algorithm. To further support such efforts, this appendix provides a few implementation details that we believe are important.

The version of UDF used for all the experiments presented in this document employs Kernel Density Estimation (KDE) for estimating the probability density function over the feature space. When using large data sets, KDE is a rather expensive method for estimating densities. Assuming that we have an data set S_L of N examples selected for labeling, and a set of M unlabeled data points to choose from, identifying the unlabeled example in the least densely populated region of the feature spaces requires $O(NM)$ operations.

Even for our smaller, fully labeled experiments M is on the order of 10^6, since M is the number of patches, which is the number of patches per image (1200 in our case) multiplied by the number of images. In this case, the number N of patches representing the data set selected for labeling has a huge impact on the overall speed of the UDF algorithm. What makes this problem worse is the data block constraint discussed in Section 2 of Chapter 3: we are not actually adding individual patches to the set S_L, but entire images. As a result, N does not grow incrementally but in large jumps. In initial implementations our algorithm would get very slow very quickly, which made it unusable for the large number of experiments we performed.

The solution employed was to limit the number of image patches that get added to the set S_L. Rather than have all or a constant fraction of the patches in each newly selected image added, we adjust the number of patches based on their likelihood under the pdf estimated using the the current set S_L: the image patches whose likelihood score falls in the bottom fraction f of the likelihood range get added to the set. Since the likelihood of the interesting patches is much lower than the average score of the image, the exact choice for the fraction f is not very important: only the most interesting patches get added to the data set S_L, which makes N grow much slower while retaining the most salient image patches from the newly selected image. In all our experiments, we have used a fraction $f = 0.5$. While the fact that the average time required for selecting each image for labeling changes based on how many images we want to select makes it difficult to provide precise numbers, selecting 15-20 images from a set of 1000 unlabeled images results in approximately 3-4 minutes per image on a 1.2Gz PentiumM computer.

This solution works relatively well. In future work we intend to replace KDE with other more efficient approximate methods for determining the density of data points in a region of the feature space. In particular, we believe that using the average distance to the k-nearest neighbors[1] might be a good enough approximation for our purposes.

[1]which can be computed much faster

REFERENCES

[1] Naoki Abe and Hiroshi Mamitsuka. Query learning strategies using boosting and bagging. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 1–10. Morgan Kaufmann Publishers, July 24-27 1998.

[2] Shlomo Argamon-Engelson and Ido Dagan. Committee-based sample selection for probabilistic classifiers. *Journal of Artificial Intelligence Research*, 11:335–360, 1999.

[3] E.B. Baum. Neural net algorithms that learn in polynomial time from examples and queries. In *IEEE Transactions on Neural Networks*, volume 2, pages 5–19, January 1991.

[4] E.B. Baum and K. Lang. Query learning can work poorly when human oracle is used. In *International Joint Conference in Neural Networks*, 1992.

[5] S. Bay, K. Saito, N. Ueda, and P. Langley. A framework for discovering anomalous regimes in multivariate time-series data with local models. `http://www.isle.org/~sbay/papers/darts.pdf`, 2005.

[6] P. Belluta, R. Manduchi, L. Matthies, K. Owens, and A. Rankin. Terrain perception for DEMO III. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 326–331, October 2000.

[7] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[8] Avrim Blum and Shuchi Chawla. Learning from labeled and unlabeled data using graph mincuts. In *Proc. 18th International Conf. on Machine Learning*, pages 19–26. Morgan Kaufmann, San Francisco, CA, 2001.

[9] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *COLT: Proceedings of the Workshop on Computational Learning Theory*, pages 92–100. Morgan Kaufmann, San Francisco, CA, 1998.

[10] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[11] David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. Active learning with statistical models. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 705–712. The MIT Press, 1995.

[12] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis And Machine Intelligence*, 24(5):603–619, 2002.

[13] M. Daily, J. Harris, D. Keirsey, K. Olin, D. Payton, K. Reiser, J. Rosenblatt, D. Tseng, and V. Wong. Autonomous cross-country navigation with the ALV. In *Proceedings of the International Conference on Robotics and Automation*, pages 718–726, 1988.

[14] M.J. Daily, J.G. Harris, and K. Reiser. An operational perception system for cross-country navigation. In *Proceedings IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 794–802, 5-9 June 1988.

[15] Ian Davis and Anthony Stentz. Sensor fusion for autonomous outdoor navigation using neural networks. In *Proceedings of the IEEE International Conference On Intelligent Robotic Systems*, volume 3, pages 338–343, August 1995.

[16] A.R. de Saint Vincent. A 3-D perception system for the mobile robot HILAIRE. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1105–1111, San Francisco, 1986.

[17] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977.

[18] Cristian Dima, Martial Hebert, and Anthony Stentz. Enabling learning from large datasets: Applying active learning to mobile robotics. In *Proceedings of the International Conference on Robotics and Automation*, volume 1, pages 108 – 114. IEEE, April 26 - May 1 2004.

[19] Cristian Dima, Nicolas Vandapel, and Martial Hebert. Classifier fusion for outdoor obstacle detection. In *International Conference on Robotics and Automation*. IEEE, April 2004.

[20] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. John Wiley and Sons, Inc., 2nd edition edition, 2001.

[21] Eleazar Eskin. Detecting errors within a corpus using anomaly detection. In *Proc. of 2000 North American Chapter of the Association of Computational Linguistics (NAACL-2000)*, Seattle, April 29–May 4, 2000.

[22] Tom Fawcett. ROC graphs: Notes and practical considerations for researchers. Technical report, HP Laboratories, Palo Alto, CA, March 2004.

[23] S. Fish. DARPA UGCV and PerceptOR program progres. In *Proceedings of SPIE Vol. #47, Battlespace Digitization and Network Centric Warfare II*, 2002.

[24] S. Fish. Overview of UGCV and PerceptOR status. In *Proceedings of SPIE Vol. #5083, Unmanned Ground Vehicle Technology V*, 2003.

[25] Scott Fish. DARPA FCS unmanned ground vehicle research initiatives. In *Proceedings of SPIE Vol. 4715, Unmanned Ground Vehicle Technology IV*, volume 4715, pages 107–111, 2002.

[26] David A. Forsyth and Jean Ponce. *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference, 2002.

[27] Yoav Freund, H.Sebastian Seung, Eli Shamir, and Naftali Tishby. Selective sampling using the query by committee algorithm. *Machine Learning*, 28:133–168, 1997.

[28] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting. Technical report, Stanford University, 1998.

[29] Rayid Ghani, Rosie Jones, and Chuck Rosenberg, editors. *The Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*, volume ICML Workshop, Washington, D.C., August 2003.

[30] Y. Goto and A. Stentz. The CMU system for mobile robot navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 99–105, Raleigh, North Carolina, 1987.

[31] Alexander Gray and Andrew Moore. Rapid evaluation of multiple density models. In *Artificial Intelligence and Statistics*, 2003.

[32] J.A. Hanley and B.J. McNeil. The meaning and use of the area under the receiver operating characteristic (ROC) curve. *Radiology*, 143(1):29–36, April 1982.

[33] Scott Y. Harmon. The ground surveillance robot (GSR): An autonomous vehicle designed to transit unknown terrain. *IEEE Journal of Robotics and Automation*, RA-3(3):266–279, June 1987.

[34] S.Y. Harmon, G.L. Bianchini, and B.E. Pinz. Sensor data fusion through a distributed blackboard. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1449–1454, San Francisco, 1986.

[35] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer-Verlag, 2001.

[36] David Haussler, Michael Kearns, and Robert Schapire. Bounds on the sample complexity of bayesian learning using information theory and the VC dimension. *Machine Learning*, 14:83–113, 1994.

[37] M. Hebert and N. Vandapel. Terrain classification techniques from ladar data for autnomous navigation. 2003.

[38] Tsai Hong Hong, Christopher Rasmussen, Tommy Chang, and Michael Shneier. Road detection and tracking for autonomous mobile robots. volume 4715, pages 311–319. SPIE, 2002.

[39] Andres Huertas, Larry Matthies, and Arturo Rankin. Stereo-vision based tree traversability analysis for autonomous off-road navigation. In *Proceedings of the IEEE Workshop on Applications of Computer Vision*, Breckenridge, Colorado, January 2005.

[40] Carl D. Crane III, David G. Armstrong II, Maryum Ahmed, Sanjay Solanki, Donald MacArthur, Erica Zawodny, Sarah Gray, Thomas Petroff, Mike Grifis, and Carl Evans. Development of an integrated sensor system for obstacle detection and terrain evaluation for application to unmanned ground vehicles. volume 5804, pages 156–165. SPIE, 2005.

[41] Thorsten Joachims. Text categorization with support vector machines: learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398, pages 137–142, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE.

[42] Thorsten Joachims. Transductive inference for text classification using support vector machines. In Ivan Bratko and Saso Dzeroski, editors, *Proceedings of ICML-99, 16th International Conference on Machine Learning*, pages 200–209, Bled, SL, 1999. Morgan Kaufmann Publishers, San Francisco, US.

[43] Paul Komarek. *Logistic Regression for Data Mining and High-Dimensional Classification*. PhD thesis, Pittsburgh, PA, May 2004.

[44] T. Lane and C. E. Brodley. An application of machine learning to anomaly detection. In *Proc. 20th NIST-NCSC National Information Systems Security Conference*, pages 366–380, 1997.

[45] David D. Lewis and William A. Gale. A sequential algorithm for training text classifiers. In *Proceedings of the 17th annual ACM SIGIR conference on Research and development in information retrieval*, pages 3–12. Springer-Verlag New York, Inc., 1994.

[46] Roberto Manduchi. Bayesian fusion of color and texture segmentations. In *Proceedings of the International Conference of Computer Vision*, 1999.

[47] L. Mason, J. Baxter, P. Bartlett, and M. Frean. Boosting algorithms as gradient descent. In S.A. Solla, T.K. Leen, and K.-R. Muller, editors, *Advances in Neural Information Processing Systems*, volume 12, pages 512–518. MIT Press, 2000.

[48] Larry Matthies and Arturo Rankin. Negative obstacle detection by thermal signature. In *Proceedings of the IEEE Conference on Intelligent Robots and Systems*, October 2003.

[49] Andrew McCallum and Kamal Nigam. Employing EM and pool-based active learning for text classification. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 359–367, 1998.

[50] Tom Minka. Judging significance from error bars. `http://research.microsoft.com/~minka/papers`, November 2002.

[51] Tom M. Mitchell. Generalization as search. *Artificial Intelligence*, 18(2), 1982.

[52] Andrew Moore. Acquisition of dynamic control knowledge for a robotic manipulator. In Michael B. Morgan, editor, *Proceedings of the 7th International Conference on Machine Learning*, pages 244–252, 340 Pine Street, 6th Fl., San Francisco, CA 94104, June 1990. Morgan Kaufmann.

[53] Andrew Moore. Efficient memory-based learning for robot control, October 1990.

[54] Andrew Moore. Very fast EM-based mixture model clustering using multiresolution KD-trees. In M. Kearns and D. Cohn, editors, *Advances in Neural Information Processing Systems*, pages 543–549, San Francisco, CA, April 1999. Morgan Kaufman.

[55] Andrew Moore and Jeff Schneider. Memory-based stochastic optimization. In D. Touretzky, M. Mozer, and M. Hasselm, editors, *Neural Information Processing Systems 8*, volume 8, pages 1066–1072. MIT Press, 1996.

[56] Andrew Moore, Jeff Schneider, Justin Boyan, and Mary Soon Lee. Q2: Memory-based active learning for optimizing noisy continuous functions. In J. Shavlik, editor, *Proceedings of the Fifteenth International Conference of Machine Learning*, pages 386–394, 340 Pine Street, 6th Fl., San Francisco, CA 94104, 1998. Morgan Kaufmann.

[57] Hans P. Moravec. The Stanford Cart and the CMU Rover. In *Proceedings of the IEEE*, volume 71, pages 872–884, 1983.

[58] David .G. Morgenthaler, Steven J. Hennessy, and Daniel Dementhon. Range-video fusion and comparison of inverse perspective algorithms in static images. *IEEE Transactions on Systems, Man and Cybernetics*, 20(6):1301–1312, November/December 1990.

[59] Ion Muslea, Steven Minton, and Craig A. Knoblock. Selective sampling with redundant views. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI)*, pages 621–626, Austin, Texas, July 30 - August 3 2000.

[60] Ian Nabley. *NETLAB: Algorithms for Pattern Recognition*. Advances in Pattern Recognition. Springer-Verlag, 2002.

[61] Kamal Nigam, Andrew K. McCallum, Sebastian Thrun, and Tom M. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3):103–134, 2000.

[62] Mark Ollis. Bayesian learning with ladar and radar sensors. Presented at the DARPA PerceptOR Workshop in Arlington, VA, January 2004.

[63] Mark Ollis and Todd Jochem. Structural method for obstacle detection and terrain classification. volume 5083, pages 1–12. SPIE, 2003.

[64] Dan Pelleg and Andrew Moore. Active learning for anomaly and rare-category detection. In *Advances in Neural Information Processing Systems 18*, December 2004.

[65] Fernando C. N. Pereira, Naftali Tishby, and Lillian Lee. Distributional clustering of English words. In *Meeting of the Association for Computational Linguistics*, pages 183–190, 1993.

[66] Robert Pless, John Larson, Scott Siebers, and Ben Westover. Evaluation of local models of dynamic backgrounds. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2003.

[67] Dean Pomerleau. Progress in neural network-based vision for autonomous robot driving. In *Proceedings of the Intelligent Vehicles '92 Symposium*, pages 391–396, 1992.

[68] Dean Pomerleau. *The Handbook of Brain Theory and Neural Networks*, chapter Neural Network Vision for Robot Driving. MIT Press, 1995.

[69] Dean Pomerleau. RALPH: Rapidly adapting lateral position handler. In *IEEE Symposium on Intelligent Vehicles*, pages 506–511, September 1995.

[70] Arturo Rankin, Andres Huertas, and Larry Matthies. Evaluation of stereo vision obstacle detection algorithms for off-road autonomous navigation. In *Proceedings of the 32nd AUVSI Symposium on Unmanned Systems*, Baltimore, June 2005.

[71] Arturo Rankin, Larry Matthies, and Andres Huertas. Daytime water detection by fusing multiple cues for autonomous off-road navigation. In *Proceedings of the 24th Army Science Conference*, Orlando, Florida, November 2004.

[72] Christopher Rasmussen. Combining laser range, color and texture cues for autonomous road following. In *Proceedings of the 2002 IEEE International*

Conference of Robotics and Automation, pages 4320–4325, Washington, D.C., May 2002.

[73] Mark Rosenblum. Immune systems are not just for making you feel better: they are for controlling autonomous robots. volume 5804, pages 191–202. SPIE, 2005.

[74] Mark Rosenblum and Benny Gothard. A high fidelity multi-sensor scene understanding system for autonomous navigation. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 637–643, October 2000.

[75] Mark Rosenblum and Venkataramanan Rajagopalan. The road more traveled: a foundation for autonomous roadway operations. volume 5804, pages 727–740. SPIE, 2005.

[76] Nicholas Roy and Andrew McCallum. Toward optimal active learning through Monte Carlo estimation of error reduction. In *Proceedings of the International Conference on Machine Learning*, June 2001.

[77] Alok Sarwal, Chris Baker, and Mark Rosenblum. Terrain classification for a ugv. volume 5804, pages 227–234. SPIE, 2005.

[78] Alok Sarwal, David Simon, and Venkat Rajagopalan. Terrain classification. volume 5083, pages 156–163. SPIE, 2003.

[79] Robert E. Schapire. The boosting approach to machine learning. MSRI Workshop on Nonlinear Estimation and Classification, 2002.

[80] Jeff Schneider and Andrew Moore. Active learning in discrete input spaces. In *Proceedings of the 34th Interface Symposium*, 2002.

[81] H.S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proceedings of the Fifth Workshop on Computaional Learning Theory*, pages 287–294, San Mateo, CA, 1992.

[82] S.A. Shafer, A. Stentz, and C.E. Thorpe. An architecture for sensory fusion in a mobile robot. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2002–2011, San Francisco, 1986.

[83] Charles M. Shoemaker and Jonathan A. Bornstein. The Demo III UGV program: A testbed for autonomous navigation research. In *Proceedings of the 1998 IEEE ISIC/CIRA/ISAS Joint Conference*, pages 644–651, Gaithersburg, MD, 1998.

[84] A. Stentz, A. Kelly, P. Rander, H. Herman, O. Amidi, R. Mandelbaum, G. Salgian, and J. Pedersen. Real-time, multi-perspective perception for unmanned ground vehicles. In *AUVSI*, 2003.

[85] A. Talukder, R. Manduchi, A. Rankin, and L. Matthies. Fast and reliable obstacle detection and segmentation for cross-country navigation. In *Proc. IEEE Intelligent Vehicles Conference 2002*, France, June 18-22 2002.

[86] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. In *Proceedings of the Seventeenth International Conference on Machine Learning*, July 2000.

[87] Nicolas Vandapel, Daniel Huber, Anuj Kapuria, and Martial Hebert. Natural terrain classification using 3-d ladar data. In *IEEE International Conference on Robotics and Automation*, April 2004.

[88] V.N. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

[89] Carl Wellington and Anthony Stentz. Learning predictions of the load-bearing surface for autonomous rough-terrain navigation in vegetations. In *Proceedings of the International Conference on Field and Service Robotics*, July 2003.

[90] Gary Witus, Robert Karlsen, James Overholt, and Grant Gerhart. Forecasting off-road trafficability from terrain appearance. volume 5804, pages 217–226. SPIE, 2005.

[91] Weng-Keen Wong, Andrew Moore, Gregory Cooper, and Michael Wagner. Rule-based anomaly pattern detection for detecting disease outbreaks. In *Eighteenth national conference on Artificial intelligence*, pages 217–223, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence.

[92] Brian Yamauchi. The Wayfarer modular navigation payload for intelligent robot infrastructure. volume 5804, pages 85–96. SPIE, 2005.

Document Log:

Manuscript Version

Revision: 1.47

—

Date: 2006/05/15 10:28:59

CRISTIAN DIMA

E-mail address: csd@cmu.edu