Learning Long-Range Vision for an Offroad Robot

by

Raia Thaïs Hadsell

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

Courant Institute of Mathematical Sciences

New York University

September 2008

Yann LeCun

ACKNOWLEDGMENTS

I have enjoyed the support, companionship, and collaboration of many people during my years at NYU. First I must acknowledge Yann LeCun, whose wisdom, generosity, and guidance has made this journey rich and meaningful. I know that I have profited from each and every argument I've lost with Yann - and there have been many.

The LAGR program has inspired, and funded, much of my research. For that I must credit the support of DARPA. The LAGR program not only provided funding, however, it also posed a difficult problem and provided a platform for research. The members of the Net-Scale/NYU LAGR team have been my conspirators and compatriots on this problem for the last three years, and our eventual successes were only possible through the joint work of all. I'd like to acknowledge Jan, Ayse, Pierre, Marco, Chris, Matt, and Koray for their contributions to the project, as well as our fearless PI's, Urs and Yann. With these people, I experienced the challenge, excitement, and exhilaration of LAGR ... as well as the frustration, despair, and frozen fingers. Long live the ticks of Monmouth County!

Sumit has been a gracious friend, officemate, and Lush compatriot for all 5 years of my time at NYU. I sincerely appreciate his friendship and support, as well as the friendship of others in CBLL and the MRL.

My parents have always encouraged me to be exactly who I am, and I thank them for that. I am grateful for their (possibly infinite) wisdom, their career advice and technical support, their love and grandparenting.

And, lastly, I acknowledge my family: Mark, Zöe, and Leif. They gave me laughter and light when the world looked dark, and Legos when I was weary.

ABSTRACT

Teaching a robot to perceive and navigate in an unstructured natural world is a difficult task. Without learning, navigation systems are short-range and extremely limited. With learning, the robot can be taught to classify terrain at longer distances, but these classifiers can be fragile as well, leading to extremely conservative planning. A robust, high-level learning-based perception system for a mobile robot needs to continually learn and adapt as it explores new environments. To do this, a strong feature representation is necessary that can encode meaningful, discriminative patterns as well as invariance to irrelevant transformations. A simple realtime classifier can then be trained on those features to predict the traversability of the current terrain.

One such method for learning a feature representation is discussed in detail in this work. Dimensionality reduction by learning an invariant mapping (DrLIM) is a weakly supervised method for learning a similarity measure over a domain. Given a set of training samples and their pairwise relationships, which can be arbitrarily defined, DrLIM can be used to learn a function that is invariant to complex transformations of the inputs such as shape distortion and rotation.

The main contribution of this work is a self-supervised learning process for long-range vision that is able to accurately classify complex terrain, permitting improved strategic planning. As a mobile robot moves through offroad environments, it learns traversability from a stereo obstacle detector. The learning architecture is composed of a static feature extractor, trained offline for a general yet discriminative feature representation, and an adaptive online classifier. This architecture reduces the effect of concept drift by allowing the online classifier to quickly adapt to very few training samples without overtraining. After experiments with several different learned feature extractors, we conclude that unsupervised or weakly supervised learning methods are

necessary for training general feature representations for natural scenes.

The process was developed and tested on the LAGR mobile robot as part of a fully autonomous vision-based navigation system.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

0.1 Robot Perception: Motivation and Philosophy

All animals use multiple sensory inputs to explore and navigate their environments. Many animals rely on vision as their primary sensor but use other sensory cues for mid- and short-range perception. Vision is very powerful and dominant for these animals, including humans, but is subject to error, especially at long ranges, where recognition and range errors become significant. One finds that there is generally an inverse relationship between the range of a sensor and its reliability: long-range vision is less reliable than close-range stereo vision, which is less reliable than touch, which for most animals conveys the greatest confidence. Human infants use exploration with tongue and mouth to augment their developing vision with the certainty of touch; within the first month of life, they can identify visual characteristics learned by mouthing (Meltzoff and Borton, 1979; Kaye and Bower, 1994). Rats have a slightly different version of this sensory configuration: a rat's long-range vision is quite poor, but its sight is augmented with a powerful olfactory system and a remarkable whisker apparatus for high-confidence close-range perception (Ritt et al., 2008).

Humans, rats, and other mobile species combine their multiple sensor modalities into a unified perceptual understanding of the world. Some of the key abilities given by this sensory apparatus are *close-range obstacle avoidance*, *long-range object recognition*, *strategic planning and navigation*, *adaptive learning of novel objects and places*, and *generalization to novel environments*. Although cognitive scientists, neuroscientists, and psychologists are beginning to understand mammalian senses such as olfaction (Buck and Axel, 1991), visual perception is still largely a mystery, as are the mechanisms by which multiple senses are joined into a unified

perceptive system which is the basis for navigation-related abilities.

Autonomous, mobile, vision-based robots provide a platform for exploring vision and perception under the same constraints and objectives that an animal might face: navigation of a dynamic environment, goal-directed planning, obstacle avoidance, localization of self and objects. Achieving these objectives with a vision-based mobile robot could potentially give valuable insights into animal perception. A robot equipped with stereo cameras and short-range sensors, significant computing capability, and autonomous mobility and control could theoretically be capable of the same behaviors as a perceptive animal. The robotics and computer vision communities' struggle to achieve this level of intelligent perception in an autonomous robot underscores the difficulty of the task. Despite the shortcomings of research efforts to build such a perceptual system, however, the motivation remains compelling. If roboticists could design such a system, it would provide a means of understanding our own visual systems: how perception is shaped by the need to explore and navigate a complex world.

We believe that integrating learning with multiple sensor modalities is a fundamental part of the puzzle. Learning - continuous adaptation to the environment - is needed to mediate the contributions of the different sensory inputs and also to smoothly adapt our interpretation of the surroundings according to evidence. Humans are continuously learning, making predictions based on our perceptions, then adapting those predictions based on experience. Placed in a new environment, we might see a shape far in the distance that we can't parse - is it a path? a bench? a strangely shaped tree, a sculpture? Driven by curiosity or other motivation, we move closer until, through greater resolution and stereo perception, the unknown reveals itself. If necessary, we use other senses - touch, perhaps even smell or sound. In difficult terrain, we find uncertainty even at close range - is the overgrown path, the steep hillside, the muddy trail, traversable or not? We cautiously test our footing and make a decision. Now it is learned: the visual percept is imbued with meaning. When the ambiguous shape or terrain is seen again, at a distance or near,

it will be known.

Accordingly, we believe that learning is also critical for successful robot perception. In order to perceive obstacles and paths at long-range, a robot must make predictions by generalizing from its previous experiences. This can be accomplished by learning the visual cues of nearby objects and associating them with traversability costs or terrain categories. To assess the traversability of these nearby objects, we can use close-range, high-confidence sensory inputs: stereo vision or direct contact. Thus, as the robot explores an environment, it is continually monitoring the nearby objects and learning their appearance, then predicting the traversability of more distant regions. This approach can be called *near-to-far* learning, a paradigm in which reliable, close-range sensors are used to train a long-range classifier. Coupled with mapping and planning capabilities, this perceptual architecture can begin to imitate the sort of human-level exploration and learning that was described in the previous paragraph.

0.2 An Architecture for Robot Perception

In the previous section, we motivated and loosely described a perceptual learning framework for a vision-based mobile robot. Following the near-to-far learning approach, the robot should be capable of exploring a new environment, labeling nearby objects through a close-range sensor and training a classifier with those labels to predict the category of distant areas and objects. In this section we outline the architecture for such a system - the necessary components, connections, and constraints.

Multi-resolution framework

First, there is the requirement of fast obstacle avoidance. The long-range vision classifier can be relatively slow, because it is making predictions about areas that are far away and therefore the latency between visual percept and wheel command can be high. However, if an obstacle

is suddenly detected at close range, it needs to be avoided, which will require faster detection, planning, and controller response. To alleviate the conflict, a multiple resolution framework can be used: a fast control loop with low latency does obstacle detection and avoidance at a low image resolution, while a slow control loop with higher latency does long-range vision at full resolution. This accords with our knowledge of human vision: research has shown that human subjects focus on nearby objects much more frequently than distant areas: occasional distant gazing suffices to maintain a global trajectory, but frequent nearby gazing is necessary for obstacle avoidance (Wagner et al., 1980).

Short-range obstacle detection

The next component is a module for short-range obstacle detection. This can be used for the "fast" loop described above, but it is also needed as a source of reliable labels for the long-range classifier. Accordingly, it should be efficient, robust to noise, and work in almost any environment without regard to lighting change, terrain change, etc. A stereo-based obstacle detector fits this description.

Long-range vision

The long-range vision module is composed of several parts that make up the near-to-far learning framework. First, there is the "near" part: the close-range, reliable *supervisor module* that generates labels for the classifier. This is the short-range obstacle detector, mentioned already.

Next, there is an *online classifier* that associates visual cues from the input image with labels, then predicts the labels for distant areas. The classifier could be a simple linear regression - something that can be trained quickly, in realtime.

How are the visual cues to be represented? The visual input must be transformed into a *feature representation*, since the raw pixels from the image have too much variance from irrelevant transformations like lighting and viewpoint changes. Thus, a feature extractor that is distinct

from the realtime classifier should generate the invariant feature representation from the visual input. Feature extraction corresponds to the early vision stages in the mammalian brain: visual data is processed and transformed by the retina, LGN, and primary visual cortex (V1), resulting in a feature representation about which little is known with any certainty.

There are many options for feature extractors. Simple color histograms can provide effective features in some basic environments, but they are quickly fooled by deep shadows or monochromatic settings. Other hand-crafted feature descriptors are too expensive to compute densely across the image, and are not well-suited to natural images. A feature representation could be learned with labeled data, but there is a risk of making the features too specific to a particular set of training data. Unsupervised or weakly supervised learning could be used to learn a more general feature set.

The long-range vision classifier should produce a prediction densely across the image, so that the predictions can be used for mapping and planning. A *sliding window* approach could be used, so that at each overlapping window position on the input image, a feature vector is computed by the feature extractor, possibly labeled by the supervisor module, then sent to the classifier for training and/or prediction.

Distance normalization

One inherent difficulty of near-to-far learning is scaling. The apparent size of objects scales inversely with distance, making it extremely difficult to generalize from near to far. This is one of the reasons why most near-to-far learning approaches use simple color features rather than complex features from large windows with shape and context. The framework we propose will need a mechanism to normalize the image such that objects appear to be the same height regardless of their distance from the robot.

Mapping and planning

Predictions from the long-range module as well as observations from the short-range modules should be accumulated in a map so that planning can be done on the basis of multiple observations. The map should accommodate our confidence in our observations: things seen close by or repeatedly have a high certainty; distant things, things with conflicting evidence, or things seen rarely, should be treated with some uncertainty. Planning is obviously a critical part of a navigation system, but it is outside the scope of this work.

We have motivated our approach to perception for a vision-based mobile robot and sketched the necessary components. The remainder of this work is its main contribution: a visual perception system for an autonomous mobile robot which integrates multiple sensory cues with online learning to predict the traversability of the environment at very long distances. The long-range vision module is composed of a learned feature extractor and an adaptive self-supervised classifier. The feature extractor is trained using the DrLIM approach to learn features that are invariant to viewpoint change. The classifier outputs are accumulated in a hyperbolic-polar map and converted to traversability costs. We find that this accurate long-range perception allows the robot to navigate to a goal in an intelligent, strategic manner. The long-range classifier is tested on the LAGR platform in outdoor, offroad environments. LAGR (Learning Applied to Ground Robots) is a DARPA program that asked participants to develop learning and vision algorithms for an offroad mobile robot and required rigorous objective testing (Jackel et al., 2006).

We now turn to a survey of related work in robotics and feature extraction.

1
RELATED WORK

This chapter gives an overview of vision-based mobile robotics and approaches to feature extraction.

1.1 Survey of Vision-based Mobile Robots

Computer vision for autonomous vehicles has a long research history, and can be broadly broken into indoor and outdoor domains, with each further divided into structured and unstructured environments. Although this thesis focuses on the last category (vision in unstructured outdoor environments), there has been considerable research in structured outdoor environments that is applicable to this problem. DeSouza and Kak published a 2002 survey of research in these four areas (DeSouza and Kak, 2002). Some research that applies learning to non-vision-based navigation is also important to mention. In this section, related work is organized by its learning strategy: deterministic (no learning), supervised, and self-supervised.

1.1.1 Stereo-based Navigation

The majority of vision-based navigation systems are deterministic and use no learning strategies to estimate traversability or locate obstacles. Hand-crafted methods can be reasonable choices in applications where long-range perception is not necessary, or where the environment is static and very well-known, or where guaranteed, predictable performance is critical, such as extra-terrestrial exploration. See (Huertas et al., 2005; Pagnot and Grandjean, 1995; Rieder et al., 2002; Singh et al., 2000; Thorpe et al., 1988; Yagi et al., 2001). These techniques assume that the characteristics of obstacles and traversable regions are fixed, and therefore they cannot easily adapt to changing environments. Without learning, such systems are constrained to a limited

range of predefined environments. Many deterministic methods for vision-based navigation rely on stereo-based obstacle detection (Kelly and Stentz, 1998; Kriegman et al., 1989; Goldberg et al., 2002; Nabbe and Hebert, 2003). A stereo algorithm finds pixel disparities between two aligned images, producing a 3d point cloud. By applying heuristics to the statistics of points collected in grid cells, obstacles and ground are identified. The performance of such stereo-based methods is limited, because stereo-based distance estimation is unreliable above 10 or 12 meters (for typical camera configurations and resolutions). This may cause the system to drive as if in a self-imposed "fog", driving into dead-ends and taking time to discover distant pathways that are obvious to a human observer. Stereo algorithms also fail when confronted by repeating or overly-smooth patterns, such as tall grass, dry scrub, or smooth pavement. Significant advances to stereo-based navigation were made in the DARPA PerceptOR program, a multi-participant program for vision-based offroad robotics that preceded LAGR (Krotkov et al., 2006). Far from providing a solution to vision-based navigation, however, the PerceptOR conclusions emphasize the limitations, myopic and other, of stereo-based systems (Kelly et al., 2006).

1.1.2 Beyond Stereo: Machine Learning for Terrain Classification

Statistical learning methods for object recognition and classification opened new avenues for vision-based mobile robotics research in the 90's. Researchers such as Dean Pomerleau and Todd Jochem recognized the potential of applying neural networks to navigation problems, and learning-based mobile robotics has been an active area of study ever since. These approaches to mobile robotics are quite diverse, but many of them share the same structure: features are extracted from a visual input (or other sensor) and a classifier is trained to recognize objects or roadway or traversable surface or some other target of interest. In some cases, the feature extractor and classifier are indistinguishable; they are parts of a single architecture and are trained together, end to end.

The paradigm approach for object recognition combines a feature extractor with a classifier. The purpose of feature extraction is to reduce the data complexity while providing relevant measurements or properties of the data to the classifier. If feature extraction is done well, a *representation* of the data is created that makes the classification task easier: "true" properties and underlying relationships between data samples are made apparent. Above all, the feature representation should encode the meaningful, discriminating properties of the data while being robust to irrelevant transformations. Since feature extraction always involves a narrowing of the information content from the original input, the task of a feature extractor is to retain all relevant properties while eliminating irrelevant information. For instance, changes in illumination are generally unimportant for object classifiers, as are changes in viewpoint or perspective or scale. Of course, the specific application may determine which visual qualities are meaningful and which are meaningless: a face detection task is designed to discriminate faces from non-faces, whereas a face verification task confirms or rejects the *identity* of a specific image of a face. A traditional object classifier generally ignores irrelevant background patterns and distractor objects in cluttered input images, seeking to extract features only from an isolated, segmented target object. In fact, the NORB dataset was constructed precisely to test object recognition systems' ability to classify images in the presence of distractors, clutter, and background noise (LeCun et al., 2004). A scene classifier, on the other hand, considers all objects and details of the image in order to classify the scene (e.g., office, field, kitchen, urban) (Murphy et al., 2003). Of course, a scene classifier must still disregard irrelevant information such as illumination changes.

1.1.3 Non-Adaptive Supervised Systems

Many systems that incorporate supervised learning methods have been proposed for structured environments, mainly for road-following applications. ALVINN by (Pomerleau, 1989; Pomerleau, 1993), MANIAC by (Jochem et al., 1995), and DAVE by (LeCun et al., 2005) are all

navigation systems trained end-to-end using human supervision. ALVINN (Autonomous Land Vehicle in a Neural Network) trained a neural network to follow roads and was successfully deployed at highway speed in light traffic. MANIAC was also a neural net based road-following navigation system. DAVE, for unstructured environments, used end-to-end learning to map visual input to steering angles, producing a system that could avoid obstacles in off-road settings but did not have the capability to navigate to a goal or map its surroundings. More recently, Andrew Ng designed an obstacle avoidance system that used supervised learning on hand-labeled monocular images and synthetic data to train an obstacle detector (Michels et al., 2005). Many other systems have been proposed in recent years that include supervised classification (Manduchi et al., 2003; Hong et al., 2002; Hamner et al., 2006; Vandapel et al., 2004; Pantofaru et al., 2003). These systems were trained offline using hand-labeled data, thus limiting the scope of their expertise to environments similar to those seen during training. In order to have reliable performance on a wide range of environments, the human-supervised training burden becomes substantial. Moreover, given the vast diversity of outdoor terrestrial environments - especially offroad environments - combined with the potential variation in weather, season, and obstacles, it is unlikely that any non-adaptive system can perform reliably at all times.

1.1.4 Adaptive Learning for Vision-Based Robotics

It seems that a truly robust vision-based navigation system must possess the ability to continuously adapt and learn. Even humans, with vastly powerful visual systems and extremely rich knowledge bases, are sometimes fooled by new environments, strange objects, and visual tricks, which cause us to re-assess and adapt to a new scene or terrain. The machine learning approach is to apply online learning, to continuously adapt a set of parameters to fit the current inputs.

1.1.5 Near-to-Far Learning

More recently, *self-supervised* systems have been developed that reduce or eliminate the need for hand-labeled training data, thus gaining flexibility in unknown environments. With self-supervision, a reliable module that determines traversability can provide labels for inputs to another classifier. This is known as *near-to-far* learning. Using this paradigm, a classifier with broad scope and range can be trained online using data from the reliable sensor (such as ladar or stereo). Not only is the burden of hand-labeling data relieved, but the system can robustly adapt to changing environments. Many systems have successfully employed near-to-far learning for color features, primarily by identifying ground patches or pixels, building color histograms, and then clustering the entire input image.

The near-to-far strategy has been used successfully for autonomous vehicles that must follow a road. In this task, the road appearance has limited variability, so simple color/texture based classifiers can often identify road surface well beyond sensor range. Using this basic strategy, self-supervised learning helped win the 2005 DARPA Grand Challenge: the winning approach used a mixture of Gaussians model to identify road surface based on color histograms extracted immediately ahead of the vehicle as it drives (Dahlkamp et al., 2006; Thrun et al., 2006). The output of the online classifier was not reliable enough to be used for navigation purposes, however, but only to modulate the speed of the vehicle. In another approach by Thrun et al., previous views of the road surface are computed using reverse optical flow, then road appearance templates are learned for several target distances (Leib et al., 2005).

Several other approaches have followed the self-supervised, near-to-far learning strategy. Stavens and Thrun used self-supervision to train a terrain roughness predictor (Stavens and Thrun, 2006). An online probabilistic model was trained on satellite imagery and ladar sensor data for the Spinner vehicle's navigation system (Sofman et al., 2006). Similarly, online self-supervised

learning was used to train a ladar-based navigation system to predict the location of a load-bearing surface in the presence of vegetation (Wellington and Stentz, 2004). A system that trains a pixel-level classifier using stereo-derived traversability labels was proposed by Ulrich (Ulrich and Nourbakhsh, 2000).

Not surprisingly, the greatest similarity to our proposed method can be found in the research of other LAGR participants. Since the LAGR program specifically focused on learning and vision algorithms that could be applied in new, never-seen terrain, using near-to-far self-supervised learning was a natural choice. Angelova et al. use self-supervised learning to train a feature extractor and classifier to discriminate terrain types such as gravel, asphalt, and soil. The visual representation is a 15 bin histogram of "texton" matches (Angelova et al., 2007). The SRI LAGR system used fast stereo and color-based online learning (Konolige et al., 2008). Staying within the realm of simple color-based near-to-far learning, Grudic and Mulligan explore the use of distance metrics for clustering traversable pixels in (Grudic and Mulligan, 2006). The Georgia Tech LAGR team built a self-supervised terrain classifier that uses traversability cues from close-range sensors (IMU, bumper switch) to train a classifier on stereo point cloud features (Kim et al., 2006). In a variation on the basic near-to-far strategy, Happold et al. use supervised learning to map stereo geometry to traversability costs; they also use self-supervised learning to map color features to stereo geometry. Thus their obstacle detection algorithm is in two phases: first color information is extracted and stereo geometry is predicted, then the predicted geometry is mapped to a traversability cost (Happold et al., 2006).

Our approach is distinguished from these by our careful examination of feature extraction methods, and our use of large image patches rather than color histograms or texture gradients or geometry statistics from stereo. Our system classifies the traversability of the image out to the horizon, unlike other, shorter-range approaches. Feature extraction is a critical component of a system that classifies large, distant image patches. If the feature representation can be found that

is invariant to irrelevant transformations of the input while remaining discriminative to terrain and obstacle types, the classifier's work is greatly reduced. We turn now to a survey of possible feature extraction methods.

1.2 A Survey of Methods for Feature Extraction

Many methods for feature extraction in images have been proposed. Our choice of a method or methods is constrained in several ways. First, realtime processing on the robot demands a feature representation that is fast to compute. Second, the features must be computed densely across the image, since the estimated costs will be used for mapping and planning. Third, the representation must be invariant (as much as possible) to scale, viewpoint, lighting, and shift, since the task of a near-to-far classifier is to generalize from near-range terrain to similar terrain that is at a distance.

1.2.1 Hand-Crafted Descriptors

Color histograms are arguably the simplest sort of feature representation, created by a discretization of a given color space (Novak and Shafer, 1992). Pixels from a color image patch are counted in an n-dimensional histogram, with bins that correspond to set color ranges. Shape and texture information are not retained in a color histogram, and the representation is very fragile to illumination changes. Such a brittle method is not appropriate for a near-to-far classifier.

The scale-invariant feature transform, or SIFT descriptor, has enjoyed wide success both academically and commercially (Lowe, 1999; Lowe, 2004). The algorithm begins with detection of scale-invariant interest points by identifying minima and maxima in a difference of Gaussians pyramid. From a neighborhood around each interest point, intensity gradients are computed and binned into an orientation histogram, weighted by a Gaussian kernel centered on the interest

point. Orientation assignment is done to choose a canonical orientation for the patch, which gives rotation invariance to the descriptor, and the histograms are intensity normalized. SIFT descriptors are useful for a wide variety of classification and recognition tasks, and are appealing for their black-box facility of use. Unfortunately, the ease of hand-built feature descriptors comes at a price - they can be expensive to compute and have a large number of components. Because of this cost, they are often only applied at sparse *interest points* on the input image, rather than densely extracted. Also, because they are hand-built rather than learned, they are not tunable to a particular application.

The histogram of oriented gradients (HOG) descriptor (Dalal and Triggs, 2005) is similar to SIFT in that HOG computes a distribution of intensity gradients to build the descriptor. For improved performance in natural images, the histograms are contrast normalized. This is done by normalizing each local histogram v by the total intensity of a surrounding neighborhood or "block" b_v in the image:

$$g(v) = \frac{v}{\sqrt{\|b_v\|_2^2 + e^2}}$$

Unlike SIFT descriptors, HOG descriptors have a broader spatial input, finer orientation binning, and are computed densely across the image at a single scale.

The region-based context feature (RCF), another example of a hand-crafted feature descriptor, integrates regional and local features (Pantofaru et al., 2006). The *shape context* feature descriptor (Mori et al., 2005) encodes the shape of an object with a histogram of edge coordinates, where the histogram is built using uniform bins in log-polar space such that nearby sampled points are more heavily weighted than further away ones.

Gabor filters are another example of a hand-built feature representation. Gabor filters are thought to be similar to receptive fields in the mammalian primary visual cortex; the receptive field of a Gabor filter is defined by the product of a harmonic function and a Gaussian function. This unlearned feature representation is often used for texture analysis (Jain et al., 1997).

1.2.2 Supervised and Hierarchical Learning

Supervised learning can be used to train a feature extractor. Typically, supervised learning is used to train the entire architecture end-to-end, so that the feature representation and the classifier are trained together. For a self-supervised learning framework, however, the classifier needs to be trained separately from the feature extractor. Thus, one can train a supervised hierarchy end-to-end, then remove the final classification layer and use the truncated network as a feature extractor. Convolutional networks are fully supervised multi-layer learning machines that are well-suited to learning patterns in images and temporal data. They are described in detail in Section 1.3.1.

There are several other hierarchical models used for pattern recognition in images which use a feature extractor-plus-classifier framework. The "biological model" of Poggio et al. is composed of a first layer of Gabor filters, a max-pooling layer, and a second layer of randomly selected patches from the training set. No learning is applied except to train the classifier (Riesenhuber and Poggio, 1999).

Although feature extractors trained with supervision can be very discriminative, one risks being *too* discriminative and throwing away valuable information. This is a real danger if the quantity or diversity of labeled data is insufficient, a situation that often arises with real-world problems. For near-to-far learning for navigation in unstructured terrain, generalization beyond the training data is of primary importance. Accordingly, we now consider *unsupervised* and *weakly supervised* feature representations.

1.2.3 Unsupervised Feature Extractors

Most unsupervised feature extractors are trained to reconstruct the input from the feature representation, usually thought of as a *code*. After training, they can be applied to de-noising problems, missing-input problems, data compression, and data retrieval problems.

For unsupervised clustering algorithms such as *k-means*, the extracted feature, or code, for an input is simply the index of the closest prototype. K-means is a well-known algorithm for unsupervised clustering. K-means does not perform well for high-dimensional inputs because of the amount and diversity of data required to find a meaningful clustering in many dimensions.

Principal component analysis (PCA) is a method for dimensionality reduction that projects the inputs onto a linear subspace that maximizes the variance of the input data (Jolliffe, 1986). The code produced by PCA is the projection of the input.

Auto-encoder neural networks train a bottlenecked network with a reconstruction criterion and gradient optimization methods. The loss function for a narrow auto-encoder is the squared reconstruction error:

$$L(X, W_D, W_C) = \|Dec_{W_D}(Enc_{W_C}(X)) - X\|^2, \tag{1.1}$$

where Dec_{W_D} and Enc_{W_C} are different layers of the same network. These networks have at least three layers: a wide layer with more units than the number of inputs, a code layer with fewer units than the input which forces a non-linear, low-dimensional projection of the input, and another wide layer whose output matches the dimensionality of the input. A significant problem with auto-encoders is that gradient descent methods have a difficult time finding good low-dimensional representations (Hinton and Salakhutdinov, 2006; Ranzato et al., 2007a).

1.2.4 Deep Learning Architectures

The rich variability of natural data requires a highly non-linear feature representation for good pattern recognition to take place. Multi-layer non-linear architectures are one way to build a highly non-linear function of the input, since to express the same function with a single layer, many more learned parameters are required, which increases the scope of the training task substantially. However, it has been known for some time that multi-layer, i.e. *deep*, networks are

very difficult to train using standard top-down gradient descent optimization (Tesauro, 1992), since randomly-initialized narrow networks are quickly trapped in local minima.

Recent research has suggested a new paradigm for training deep architectures. Multiple layers of the network are trained separately and sequentially, using unsupervised data. Supervision can be used to "fine-tune" the learned filters. In (Hinton et al., 2006), *deep belief nets* are proposed: a multi-layer architecture that is composed of restricted Boltzmann machines, individually trained with contrastive divergence to reconstruct their input by learning the distribution of the input data. The learned layers can then be used to initialize a supervised network trained with back-propagation. By initializing the individual layers with meaningful patterns rather than random weights, the basic obstruction to training deep architectures is removed. Unsupervised training of the individual layers is important; supervised greedy training was found to degrade results, presumably because the supervised training overly restricted the information content of the initialized network (Bengio et al., 2006).

Following the same paradigm, improved classification results were obtained by training individual layers as auto-encoders, followed by supervised training (Ranzato et al., 2007c). Weston and Collobert used a weakly supervised learning criterion (the DrLIM approach) to train intermediate layers of a multi-layer network while simultaneously applying supervised top-down training, resulting in state of the art advances for multiple natural language processing tasks (Weston et al., 2008; Collobert and Weston, 2008).

1.2.5 Learned Similarity Metrics

Learning a similarity metric using neighborhood information, usually pairwise similarity labels, is a way to *weakly* supervise a feature representation. Since no targets are given in an output space, the configuration of the output manifold is almost entirely data-driven.

There are many powerful approaches to the problem of mapping a set of high-dimensional

points onto a low-dimensional manifold, including the classic linear methods: principal component analysis (Jolliffe, 1986) and multi-dimensional scaling (Cox and Cox, 1994) as well as newer, non-linear, spectral methods: ISOMAP by (Tenenbaum et al., 2000), Local Linear Embedding - LLE by (Roweis and Saul, 2000), Laplacian Eigenmaps due to (Belkin and Niyogi, 2003) and Hessian LLE by (Donoho and Grimes, 2003). However, none of these methods attempt to compute a *function* that could map a new, unknown data point without recomputing the entire embedding and without knowing its relationships to the training points, making these approaches unsuitable for feature extraction in an online setting. Out-of-sample extensions to the above methods have been proposed in (Bengio et al., 2004), but they too rely on a predetermined computable distance metric.

Most recently, new methods for pairwise supervised dimensionality reduction methods have been proposed that actually learn an embedding function, which can be used as a learned similarity metric, or as a low-dimensional feature representation. Neighbourhood component analysis (NCA) is a method that optimizes the *leave-one-out* error of k-nearest-neighbor classification (Goldberger et al., 2005). The actual leave-one-out error is replaced by a soft version to facilitate gradient optimization. The learned embedding function can be linear or non-linear, and the pairwise labeling can come from any source. Stochastic neighbour embedding is another pairwise learning approach (Hinton and Roweis, 2004). The cost function for SNE is the sum of Kullback-Liebler divergences for similar inputs. The approach can be extended to optimize multiple embeddings of each sample, allowing greater disambiguation, but this extension is only possible on the training data, not on unseen samples.

Variable-kernel similarity metric learning is a learned low-dimensional linear embedding of the input for the purpose of improved k-nearest-neighbor performance (Lowe, 1995). The linear projection is optimized with cross-validation. The algorithm uses a dynamically sized Gaussian weighting window whose width increases in neighborhoods with few training samples

and decreases in more populated neighborhoods.

DrLIM is another approach to similarity metric learning which is appropriate for learning a feature representation (Hadsell et al., 2006a). It relies only on pairwise neighbor labels rather than distances in the high-dimensional space. This approach is described in detail in Chapter 2.

1.3 Methods of Interest

Convolutional neural networks and convolutional auto-encoders are used extensively in the remainder of the thesis. Both learning architectures are described in detail in this section.

1.3.1 Convolutional Networks

In the 1960's, biologists Hubel and Wiesel proposed a functional model of cells in V1, the primary visual cortex of mammals. The model was composed of *simple* cells, which detected oriented edges, and *complex* cells, which pooled the outputs of multiple simple cells, creating invariance to spatial position (Hubel and Wiesel, 1962). The *Neocognitron* (Fukushima, 1980) was an artificial neural network architecture that was directly inspired by Hubel and Wiesel's model, containing alternating layers of local receptive fields (similar to simple cells) and pooling units (similar to complex cells).

Convolutional networks are similar to Fukushima's Neocognitron. They are multi-layer, non-linear, learning architectures that learn low-level features and high-level representations and are end-to-end trainable with gradient backpropagation (LeCun and Bengio, 1995). They are especially well-suited to vision applications, because they are naturally shift and scale invariant. A standard convolutional network is composed of two types of alternating layers: *convolutional* layers and *subsampling*, or pooling, layers. A convolutional layer contains local receptive fields that are trained to extract local features and patterns across the input. Pooling between convolu-

tional layers increases the shift and scale invariance while reducing computational complexity.

The convolutional network described in Section 2.3.1 is a typical design for pattern recognition in images. See Fig. 2.3 for a diagram of that architecture.

Convolution Layer

The convolutional layer performs a series of convolutions with the input using a set of learned filters. This can be thought of as a *sliding window pattern detector*. The basic operation is a convolution. A 2D filter is scanned over an entire input plane, and at each overlapping location a dot product is computed of the filter and the input values at that location. The dot product is the *response* of the filter at that location, and it measures the degree of correlation. The response is combined with an additive bias and "squashed" through a sigmoid function, and output to a corresponding position in a *feature map*. The function computed on input layer x and filter f and output feature map z is

$$z_j = \sigma(c_j \sum_i x_i * f_{ij} + b_j), \tag{1.2}$$

where σ is a sigmoid function, $*$ denotes the convolution operator, i indexes the input layer, j indexes the output feature map, and c_j and b_j are multiplicative and additive constants.

Since the same filter is passed over the entire input, the same pattern is being detected everywhere. Sharing weights across the entire input plane in this manner creates invariance to translation and robustness to distortion. Multiple filters extract different features, and the feature maps can be further combined in different, non-symmetric combinations, so that the number of output feature maps generally exceeds the number of input maps. The role of the convolutional layer is to extract increasingly complex features through multiple filters and multiple layers.

Pooling Layer

Pooling local features together gives invariance to exact position while retaining relative positions between features. Subsampling or max-sampling between convolutional layers increases

invariance while decreasing complexity. Pooling fields are typically 2x2 or 3x3. An average or max function is computed over the field, a trainable coefficient and bias are applied, and the result is passed through a sigmoid function.

Applications

Convolutional networks have been used for many applications. Notably, they were at the center of the highly successful handwriting recognition system introduced in 1998 (LeCun et al., 1998). They have been used for face recognition (Lawrence et al., 1997; Osadchy et al., 2007), robot vision (Happold et al., 2006), license-plate recognition, and many other tasks.

1.3.2 Convolutional Auto-Encoder

Recently, (Ranzato et al., 2007c) described an energy-based method for training an auto-encoder architecture. The learned filters can then be used to initialize a standard convolutional network, and top-down supervised learning can be applied to fine-tune the filters. This is similar to the learning approach used by (Hinton et al., 2006) to train the deep belief net. To train a single layer of filters as an auto-encoder, using the method advanced in (Ranzato et al., 2007c) and (Ranzato et al., 2007a), two functions are learned. The *encoder* $F_{enc}(X)$ takes an input X and predicts the best low-dimensional code Z. The *decoder* $F_{dec}(Z)$ tries to recreate X from a code Z. The energy of the system is the sum of two costs: the error predicting the optimal code $E_{enc}(F_{enc}, Z_{opt})$, and the error reconstructing the input $E_{dec}(F_{dec}, X)$. A non-linear *sparsifying logistic* function can also be applied within this architecture that transforms the codeword into a sparse vector.

1.4 Overview of Thesis

This thesis presents two significant contributions to research in invariant feature learning for vision-based robot navigation. In Chapter 2, a method for learning a feature representation which can incorporate complicated, highly non-linear invariances is described. The approach, dimensionality reduction by learning an invariant mapping (DrLIM), is pairwise supervised - it only needs neighborhood relations between training samples, and the relationships could come from prior knowledge, or from manual labeling, and they can be independent of any distance metric. DrLIM can be used to learn a function that is invariant to complex transformations of the inputs such as shape distortion and rotation. The basic approach is explained in Chapter 2 and results are given for various demonstrative experiments. The DrLIM approach can also be used to train a similarity metric for a particular application domain. An application of DrLIM to the problem of *face verification* is described in Chapter 3.

The latter half of the thesis describes an autonomous navigation system for an offroad mobile robot, with focus on the long-range terrain classifier. The long-range vision module uses self-supervised near-to-far learning to train a classifier in realtime to detect obstacles and paths as far as 200 meters away. This allows for strategic planning and navigation towards a distant goal. The classifier relies on a robust feature extractor, which is trained offline. We experimented extensively with different methods for feature extraction, including both supervised and unsupervised feature learning and hybrid approaches. The accuracy of the different feature representations was tested using a hand-labeled groundtruth dataset. The performance of the full navigation system was assessed through field tests and observation. The long-range vision module is detailed in Chapter 4, the long-range mapping and planning approach is described in Chapter 5, and quantitative and qualitative results are given in Chapter 6.

2

DIMENSIONALITY BY LEARNING AN INVARIANT MAPPING

Joint work with Sumit Chopra.

2.1 Introduction

Modern applications have steadily expanded their use of complex, high-dimensional data. The massive, high-dimensional image datasets generated by biology, earth science, astronomy, robotics, modern manufacturing, and other domains of science and industry demand new techniques for analysis, feature extraction, dimensionality reduction, and visualization.

Dimensionality reduction aims to translate high-dimensional data to a low-dimensional representation such that similar input objects are mapped to nearby points on a manifold. Most existing dimensionality reduction techniques have two shortcomings. First, they do not produce a *function* (or a mapping) from input to manifold that can be applied to new points whose relationship to the training points is unknown. Second, many methods presuppose the existence of a meaningful (and computable) distance metric in the input space. Our interests lie in training a low-dimensional embedding that can be used as an invariant feature representation, so learning a function that is robust to unseen data is critical.

For example, Locally Linear Embedding (LLE) (Roweis and Saul, 2000) linearly combines input vectors that are identified as neighbors. The applicability of LLE and similar methods to image data is limited because linearly combining images only makes sense for images that are well registered and very similar. Laplacian Eigenmap (Belkin and Niyogi, 2003) and Hessian LLE (Donoho and Grimes, 2003) do not require a meaningful metric in input space (they merely

require a list of neighbors for every sample), but as with LLE, new points whose relationships with training samples are unknown cannot be processed. Out-of-sample extensions to several dimensionality reduction techniques have been proposed that allow for consistent embedding of new data samples without recomputation of all samples (Bengio et al., 2004). These extensions, however, assume the existence of a computable kernel function that is used to generate the neighborhood matrix. This dependence is reducible to the dependence on a computable distance metric in input space.

Another limitation of current methods is that they tend to cluster points in output space, sometimes densely enough to be considered degenerate solutions. Rather, it is sometimes desirable to find manifolds that are uniformly covered by samples.

The method proposed here, called Dimensionality Reduction by Learning an Invariant Mapping (DrLIM), provides a solution to the above problems. DrLIM is a method for learning a globally coherent non-linear function that maps the data to a low-dimensional manifold. The method presents four essential characteristics:

- It only needs neighborhood relationships between training samples. These relationships could come from prior knowledge, or manual labeling, and be independent of any distance metric.

- It may learn functions that are invariant to complicated non-linear transformations of the inputs such as lighting changes and geometric distortions.

- The learned function can be used to map new samples not seen during training, with no prior knowledge.

- The mapping generated by the function is in some sense "smooth" and coherent in the output space.

A contrastive loss function is employed to learn the parameters W of a parameterized function G_W, in such a way that neighbors are pulled together and non-neighbors are pushed apart. Prior knowledge can be used to identify the neighbors for each training data point.

The method uses an energy based model that uses the given neighborhood relationships to learn the mapping function. For a family of functions G, parameterized by W, the objective is to find a value of W that maps a set of high-dimensional inputs to the manifold such that the Euclidean distance between points on the manifold, $D_W(\vec{X_1}, \vec{X_2}) = ||G_W(\vec{X_1}) - G_W(\vec{X_2})||_2$ approximates the "semantic similarity"of the inputs in input space, as provided by a set of neighborhood relationships. No assumption is made about G_W except that it is differentiable with respect to W.

Section 2.2 describes the general framework and the loss function. The ideas in this section are made concrete in section 2.3. Here various experimental results are given.

2.2 Learning the Low-Dimensional Mapping

The problem is to find a function that maps high-dimensional input patterns to lower dimensional outputs, given neighborhood relationships between samples in input space. The graph of neighborhood relationships may come from information source that may not be available for test points, such as prior knowledge, manual labeling, etc. More precisely, given a set of input vectors $\mathcal{I} = \{\vec{X_1}, \ldots, \vec{X_P}\}$, where $\vec{X_i} \in \Re^D, \forall i = 1, \ldots, n$, find a parametric function $G_W : \Re^D \longrightarrow \Re^d$ with $d \ll D$, such that it has the following properties:

1. Simple distance measures in the output space (such as Euclidean distance) should approximate the *neighborhood relationships* in the input space.

2. The mapping should not be constrained to implementing simple distance measures in the input space and should be able to learn invariances to complex transformations.

3. It should be *faithful* even for samples whose neighborhood relationships are unknown.

2.2.1 The Contrastive Loss Function

Consider the set $\mathcal{I}$ of high-dimensional training vectors $\vec{X}_i$. Assume that for each $\vec{X}_i \in \mathcal{I}$ there is a set $\mathcal{S}_{\vec{X}_i}$ of training vectors that are deemed similar to $\vec{X}_i$. This set can be computed by some prior knowledge - invariance to distortions or temporal proximity, for instance - which does not depend on a simple distance. A meaningful mapping from high to low-dimensional space maps similar input vectors to nearby points on the output manifold and dissimilar vectors to distant points. A new loss function whose minimization can produce such a function is now introduced.

Unlike loss functions that sum over samples, this loss function runs over *pairs of samples*. Let $\vec{X}_1, \vec{X}_2 \in \mathcal{I}$ be a pair of input vectors shown to the system. Let Y be a binary label assigned to this pair. $Y = 0$ if $\vec{X}_1$ and $\vec{X}_2$ are deemed similar, and $Y = 1$ if they are deemed dissimilar. Define the parameterized distance function to be learned D_W between $\vec{X}_1, \vec{X}_2$ as the Euclidean distance between the outputs of G_W. That is,

$$D_W(\vec{X}_1, \vec{X}_2) = \|G_W(\vec{X}_1) - G_W(\vec{X}_2)\|_2 \tag{2.1}$$

To shorten notation, $D_W(\vec{X}_1, \vec{X}_2)$ is written D_W. Then the loss function in its most general form is

$$\mathcal{L}(W) = \sum_{i=1}^{P} L(W, (Y, \vec{X}_1, \vec{X}_2)^i) \tag{2.2}$$

$$L(W, (Y, \vec{X}_1, \vec{X}_2)^i) = (1 - Y) L_S\left(D_W^i\right) + Y L_D\left(D_W^i\right) \tag{2.3}$$

where $(Y, \vec{X}_1, \vec{X}_2)^i$ is the i-th labeled sample pair, L_S is the partial loss function for a pair of similar points, L_D the partial loss function for a pair of dissimilar points, and P the number of training pairs (which may be as large as the square of the number of samples).

L_S and L_D must be designed such that minimizing L with respect to W would result in low values of D_W for similar pairs and high values of D_W for dissimilar pairs.

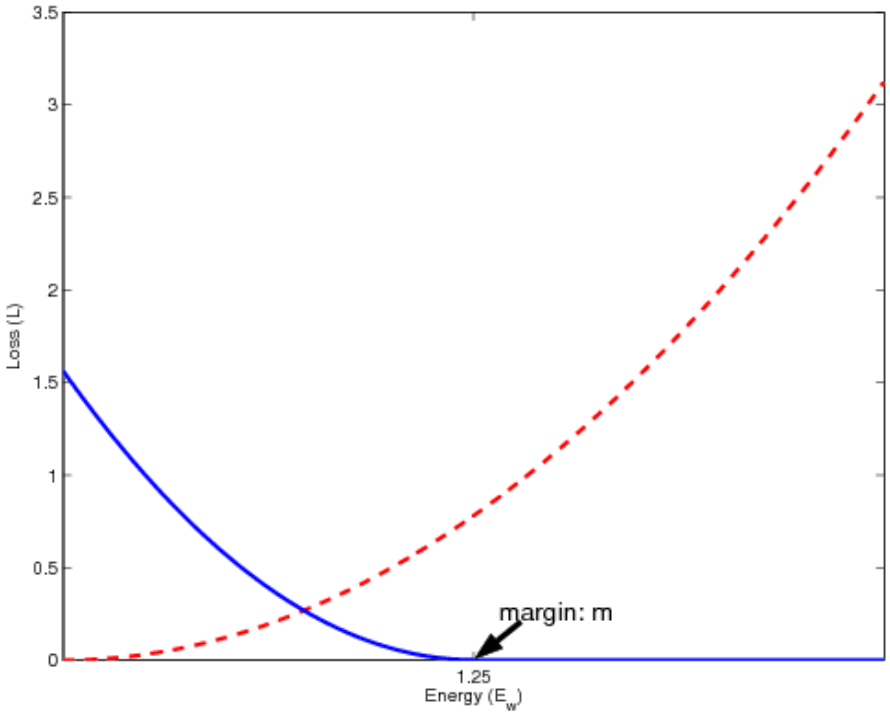


Figure 2.1: Graph of the loss function L against the energy D_W. The dashed (red) line is the loss function for the similar pairs and the solid (blue) line is for the dissimilar pairs.

The exact loss function is

$$L(W, Y, \vec{X_1}, \vec{X_2}) = (1 - Y)\frac{1}{2}(D_W)^2 + (Y)\frac{1}{2}\{max(0, m - D_W)\}^2 \tag{2.4}$$

where $m > 0$ is a margin. The margin defines a radius around $G_W(\vec{X})$. Dissimilar pairs contribute to the loss function only if their distance is within this radius (see Fig. 2.1).

The contrastive term involving dissimilar pairs, L_D, is crucial. Simply minimizing $D_W(\vec{X_1}, \vec{X_2})$ over the set of all similar pairs will usually lead to a collapsed solution, since D_W and the loss L could then be made zero by setting G_W to a constant. Most energy-based models require the use of an explicit contrastive term in the loss function.

2.2.2 The Algorithm and Learning Architecture

Learning the DrLIM loss function is realized by training a network that consists of two identical convolutional networks that share the same set of weights - a *Siamese architecture* (Bromley et al., 1993) (see Fig. 2.2). The Siamese framework comprises two identical networks and one cost module. The input to the system is a pair of images and a label. The images are passed through the sub-networks, yielding two outputs which are passed to the cost module which produces the scalar energy as discussed in section 2.2.1. The loss function combines the label with energy, and gradient-based backpropagation was used to train the system. For the function G_W we use a convolutional network (see Section 1.3.1).

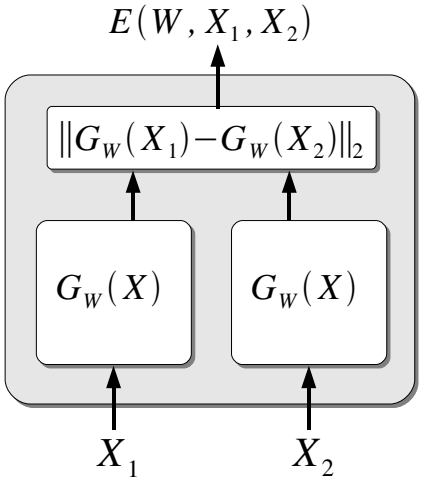


Figure 2.2: A *Siamese* architecture comprises two identical parameterized functions.

The algorithm first generates the training set, then trains the machine.

Step 1: For each input sample $\vec{X}_i$, do the following:

(a) Using prior knowledge find the set of samples $\mathcal{S}_{\vec{X}_i} = \{\vec{X}_j\}_{j=1}^{p}$, such that $\vec{X}_j$ is deemed similar to $\vec{X}_i$.

(b) Pair the sample $\vec{X_i}$ with all the other training samples and label the pairs so that: $Y_{ij} = 0$ if $\vec{X_j} \in \mathcal{S}_{\vec{X_i}}$, and $Y_{ij} = 1$ otherwise.

Combine all the pairs to form the labeled training set.

Step 2: Repeat until convergence:

(a) For each pair $(\vec{X_i}, \vec{X_j})$ in the training set, do

i. If $Y_{ij} = 0$, then update W to decrease $D_W = \|G_W(\vec{X_i}) - G_W(\vec{X_j})\|_2$

ii. If $Y_{ij} = 1$, then update W to increase $D_W = \|G_W(\vec{X_i}) - G_W(\vec{X_j})\|_2$

This increase and decrease of Euclidean distances in the output space is done by minimizing the above loss function.

2.3 Experiments

The experiments presented in this section demonstrate the invariances afforded by our approach and also clarify the limitations of techniques such as LLE. First we give details of the parameterized machine G_W that learns the mapping function.

2.3.1 Training Architecture

The learning architecture is similar to the one used in (Bromley et al., 1993) and (Chopra et al., 2005). Called a *Siamese* architecture, it consists of two copies of the function G_W which share the same set of parameters W, and a cost module. A loss module whose input is the output of this architecture is placed on top of it. The input to the entire system is a pair of images $(\vec{X_1}, \vec{X_2})$ and a label Y. The images are passed through the functions, yielding two outputs

$G(\vec{X_1})$ and $G(\vec{X_2})$. The cost module then generates the distance $D_W(G_W(\vec{X_1}), G_W(\vec{X_2}))$. The loss function combines D_W with label Y to produce the scalar loss L_S or L_D, depending on the label Y. The parameter W is updated using stochastic gradient. The gradients can be computed by back-propagation through the loss, the cost, and the two instances of G_W. The total gradient is the sum of the contributions from the two instances.

The experiments involving airplane images from the NORB dataset (LeCun et al., 2004) use a 2-layer fully connected neural network as G_W. The number of hidden and output units used was 20 and 3 respectively. Experiments on the MNIST dataset used a convolutional network as G_W (Fig. 2.3). See Section 1.3.1 for an overview of convolutional networks.

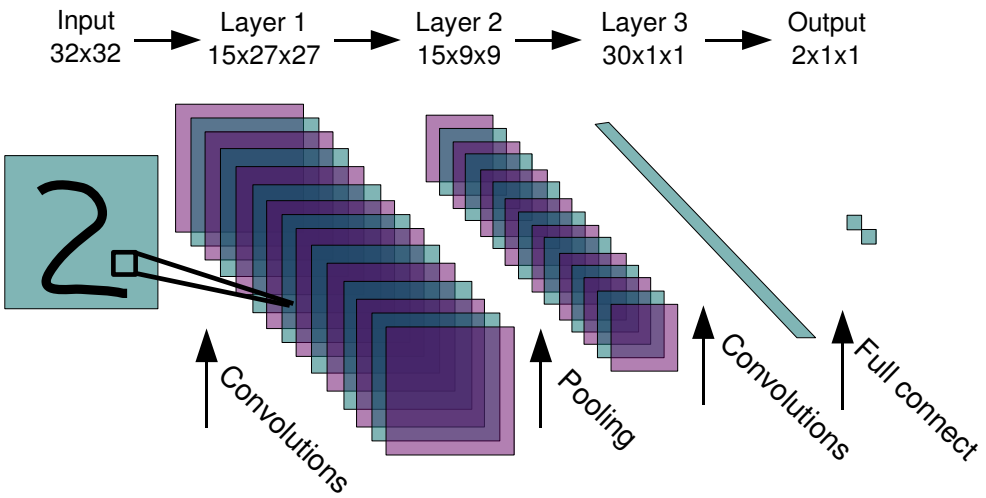


Figure 2.3: Architecture of the function G_W (a convolutional network) which was learned to map the MNIST data to a low-dimensional manifold with invariance to shifts.

The layers of the convolutional network comprise a convolutional layer C_1 with 15 feature maps, a subsampling layer S_2, a second convolutional layer C_3 with 30 feature maps, and fully connected layer F_3 with 2 units. The sizes of the filters for the C_1 and C_3 were 6x6 and 9x9 respectively.

2.3.2 Learned Mapping of MNIST Samples

The first experiment is designed to establish the basic functionality of the DrLIM approach. The neighborhood graph is generated with Euclidean distances and no prior knowledge.

The training set is built from 3000 images of the handwritten digit 4 and 3000 images of the handwritten digit 9 chosen randomly from the MNIST dataset. Approximately 1000 images of each digit comprised the test set. These images were shuffled, paired, and labeled according to a simple Euclidean distance measure: each sample $\vec{X}_i$ was paired with its 5 nearest neighbors, producing the set S_{X_i}. All other possible pairs were labeled dissimilar.

The mapping of the test set to a 2D manifold is shown in Figure 2.4. The lighter-colored blue dots are 9's and the darker-colored red dots are 4's. Several input test samples are shown next to their manifold positions. The 4's and 9's are in two somewhat overlapping regions, with an overall organization that is primarily determined by the slant angle of the samples. The samples are spread rather uniformly in the populated region.

2.3.3 Learning a Shift-Invariant Mapping of MNIST Samples

In this experiment, the DrLIM approach is evaluated using 2 categories of MNIST, distorted by adding samples that have been horizontally translated. The objective is to learn a 2D mapping that is invariant to horizontal translations.

In the distorted set, 3000 images of 4's and 3000 images of 9's are horizontally translated by -6, -3, 3, and 6 pixels and combined with the originals, producing a total of 30,000 samples. The 2000 samples in the test set were distorted in the same way.

First the system was trained using pairs from a Euclidean distance neighborhood graph (5 nearest neighbors per sample), as in experiment 1. The large distances between translated samples creates a disjoint neighborhood relationship graph and the resulting mapping is disjoint as

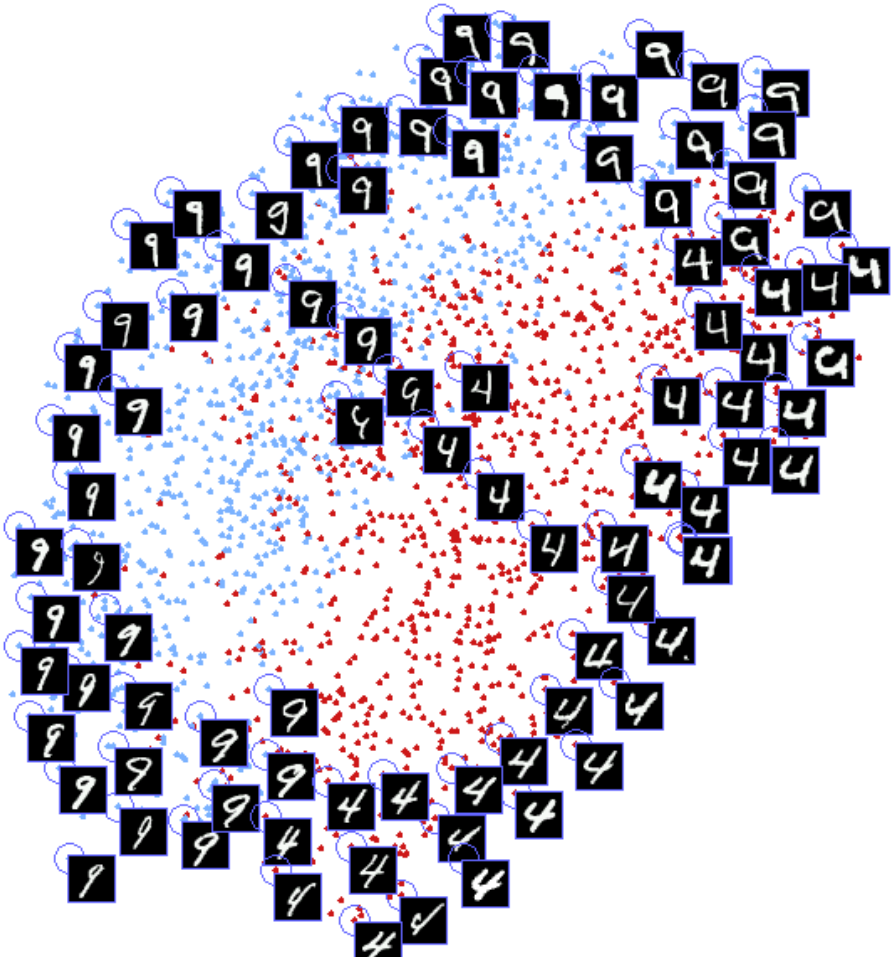


Figure 2.4: Experiment demonstrating the effectiveness of the DrLIM in a trivial situation with MNIST digits. A Euclidean nearest neighbor metric is used to create the local neighborhood relationships among the training samples, and a mapping function is learned with a convolutional network. The figure shows the placement of the *test* samples in output space. Even though the neighborhood relationships among these samples are unknown, they are well organized and evenly distributed on the 2D manifold.

well. The output points are clustered according to the translated position of the input sample (see Fig. 2.5). Within each cluster, however, the samples are well organized and evenly distributed.

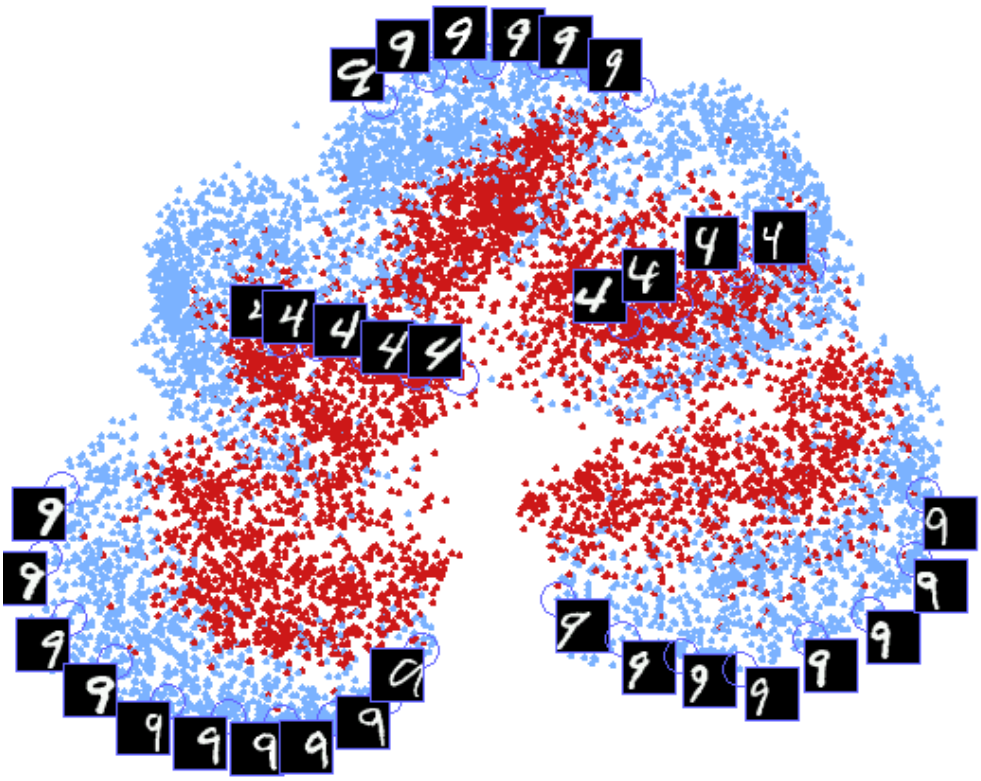


Figure 2.5: This experiment shows the effect of a simple distance-based mapping on MNIST data with horizontal translations added (-6, -3, +3, and +6 pixels). Since translated samples are far apart, the manifold has 5 distinct clusters of samples corresponding to the 5 translations. Note that the clusters are individually well-organized, however. Results are on test samples, unseen during training.

For comparison, the LLE algorithm was used to map the distorted MNIST using the same Euclidean distance neighborhood graph. The result was a degenerate embedding in which dif-

ferently registered samples were completely separated (see Fig. 2.6). Although there is sporadic local organization, there is no global coherence in the embedding.

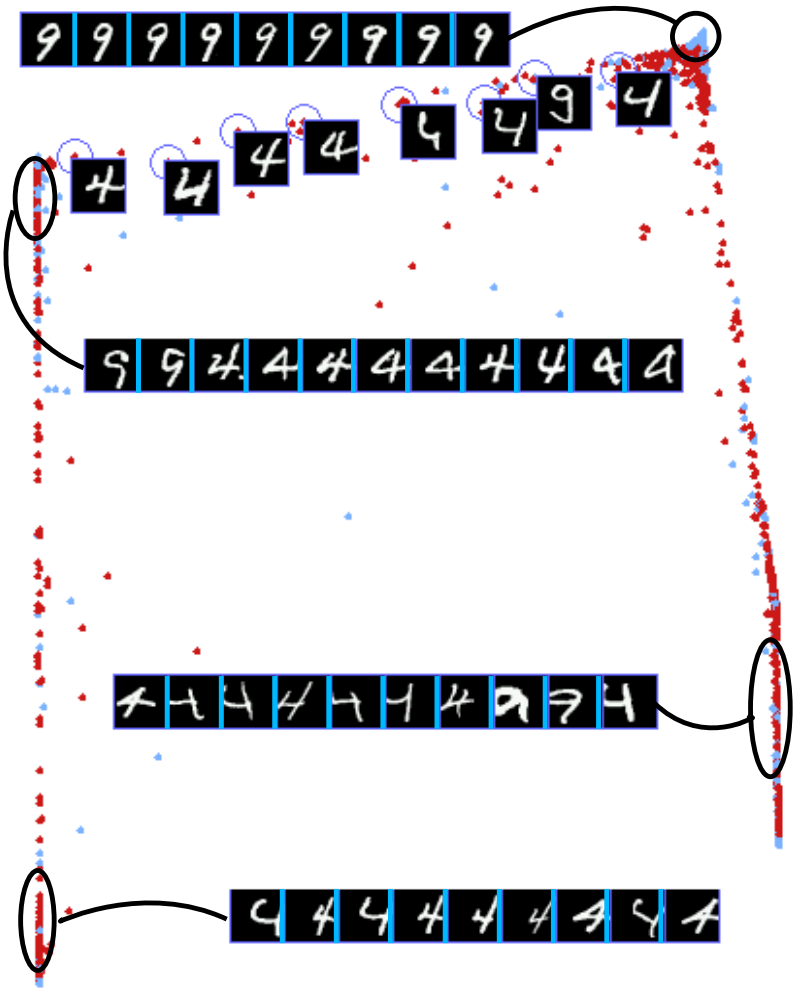


Figure 2.6: LLE's embedding of the distorted MNIST set with horizontal translations added. Most of the untranslated samples are tightly clustered at the top right corner, and the translated samples are grouped at the sides of the output.

In order to make the mapping function invariant to translation, the Euclidean nearest neighbors were supplemented with pairs created using *prior knowledge*. Each sample was paired with *(a)* its 5 nearest neighbors, *(b)* its 4 translations, and *(c)* the 4 translations of each of its 5 nearest neighbors. Additionally, each of the sample's 4 translations was paired with *(d)* all the above nearest neighbors and translated samples. All other possible pairs are labeled as dissimilar.

The mapping of the test set samples is shown in Figure 2.7. The lighter-colored blue dots are 4's and the darker-colored red dots are 9's. As desired, there is no organization on the basis of translation; in fact, translated versions of a given character are all tightly packed in small regions on the manifold.

2.3.4 Mapping Learned with Temporal Neighborhoods and Lighting Invariance

The final experiment demonstrates dimensionality reduction on a set of images of a single object. The object is an airplane from the NORB (LeCun et al., 2004) dataset with uniform backgrounds. There are a total of 972 images of the airplane under various poses around the viewing half-sphere, and under various illuminations. The views have 18 azimuths (every 20 degrees around the circle), 9 elevations (from 30 to 70 degrees every 5 degrees), and 6 lighting conditions (4 lights in various on-off combinations). The objective is to learn a globally coherent mapping to a 3D manifold that is invariant to lighting conditions. A pattern based on temporal continuity of the camera was used to construct a neighborhood graph; images are similar if they were taken from contiguous elevation or azimuth regardless of lighting. Images may be neighbors even if they are very distant in terms of Euclidean distance in pixel space, due to different lighting.

The dataset was split into 660 training images and a 312 test images. The result of training on all 10989 similar pairs and 206481 dissimilar pairs is a 3-dimensional manifold in the shape of a cylinder (see Fig. 2.8). The circumference of the cylinder corresponds to change in azimuth in input space, while the height of the cylinder corresponds to elevation in input space. The

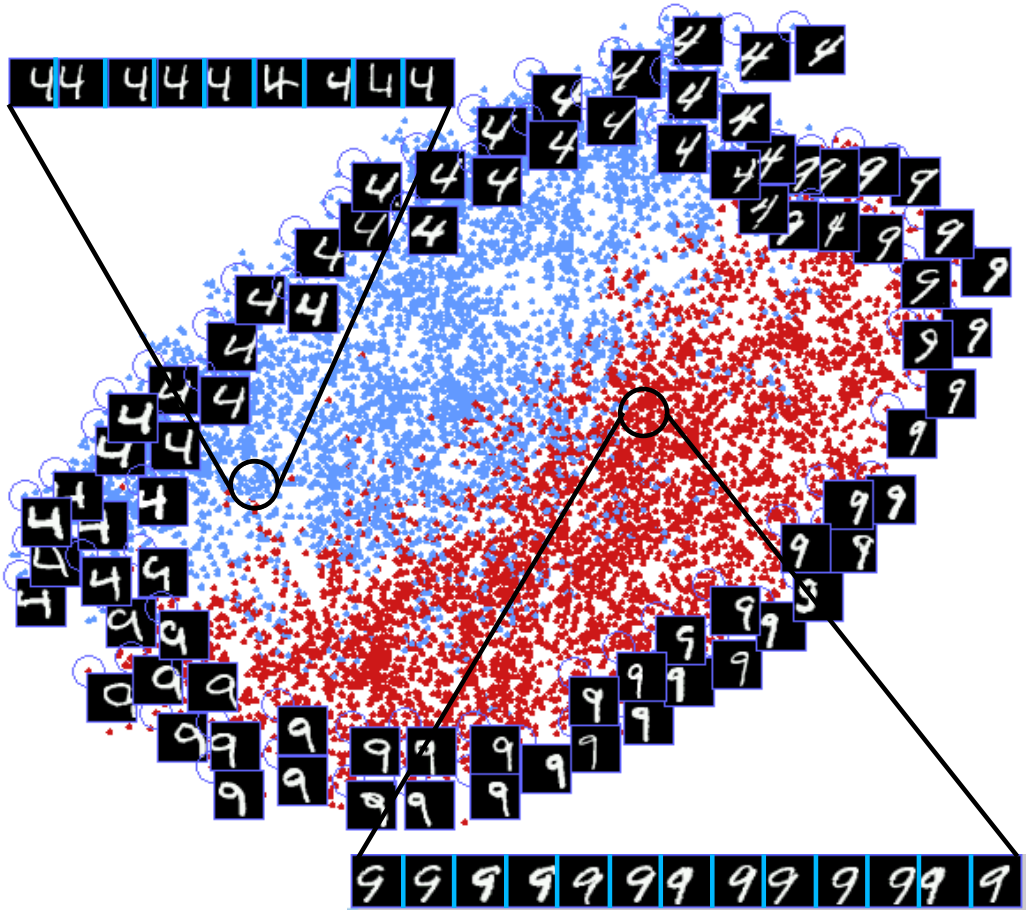


Figure 2.7: This experiment measured DrLIM's success at learning a mapping from high-dimensional, shifted digit images to a 2D manifold. The mapping is invariant to translations of the input images. The mapping is well-organized and globally coherent. Results shown are the test samples, whose neighborhood relations are unknown. Similar characters are mapped to nearby areas, regardless of their shift.

mapping is completely invariant to lighting. This outcome is quite remarkable. Using only local neighborhood relationships, the learned manifold corresponds globally to the positions of the camera as it produced the dataset.

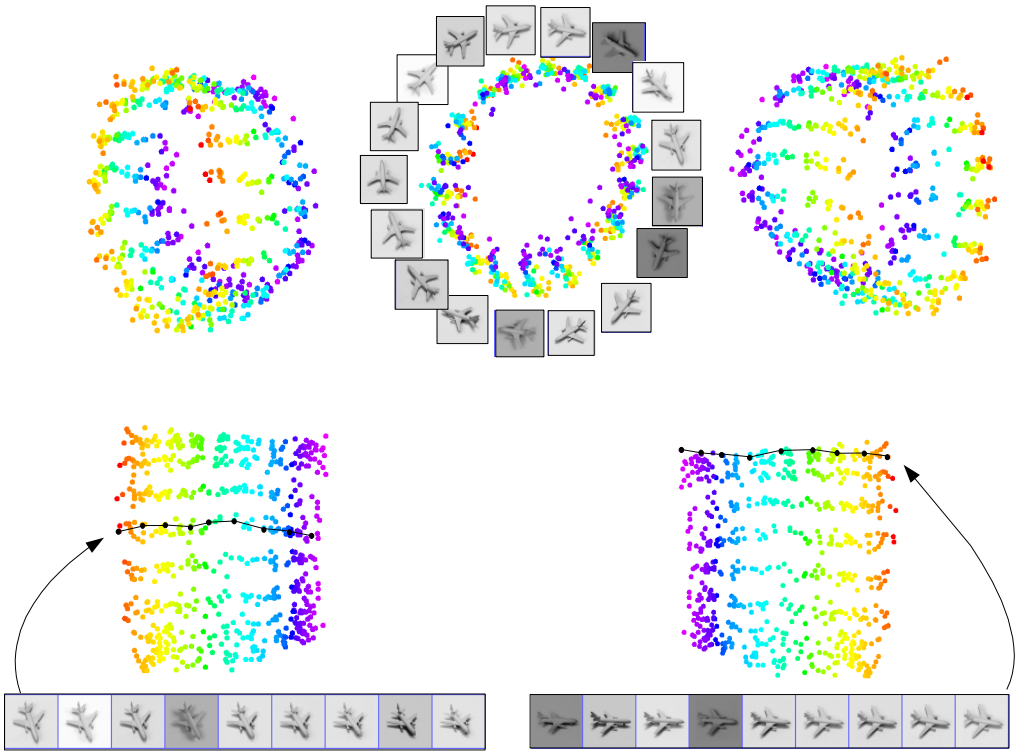


Figure 2.8: Test set results: the DrLIM approach learned a mapping to 3d space for images of a single airplane (extracted from NORB dataset). The output manifold is shown under five different viewing angles. The manifold is roughly cylindrical with a systematic organization: along the circumference varies azimuth of camera in the viewing half-sphere. Along the height varies the camera elevation in the viewing sphere. The mapping is invariant to the lighting condition, thanks to the prior knowledge built into the neighborhood relationships.

Viewing the weights of the network helps explain how the mapping learned illumination invariance (see Fig. 2.9). The concentric rings match edges on the airplanes to a particular azimuth and elevation, and the rest of the weights are close to 0. The dark edges and shadow of the wings, for example, are relatively consistent regardless of lighting.

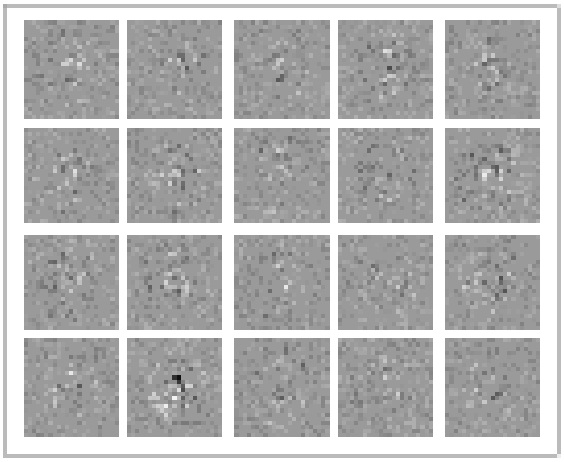


Figure 2.9: The weights of the 20 hidden units of a fully-connected neural network trained with DrLIM on airplane images from the NORB dataset. Since the camera rotates 360^o around the airplane and the mapping must be invariant to lighting, the weights are zero except to detect edges at each azimuth and elevation; thus the concentric patterns.

For comparison, the same neighborhood relationships defined by the prior knowledge in this experiment were used to create an embedding using LLE. Although arbitrary neighborhoods can be used in the LLE algorithm, the algorithm computes linear reconstruction weights to embed the samples, which severely limits the desired effect of using distant neighbors. The embedding produced by LLE is shown (see Fig. 2.10). Clearly, the 3D embedding is not invariant to lighting, and the organization of azimuth and elevation does not reflect the real topology neighborhood graph.

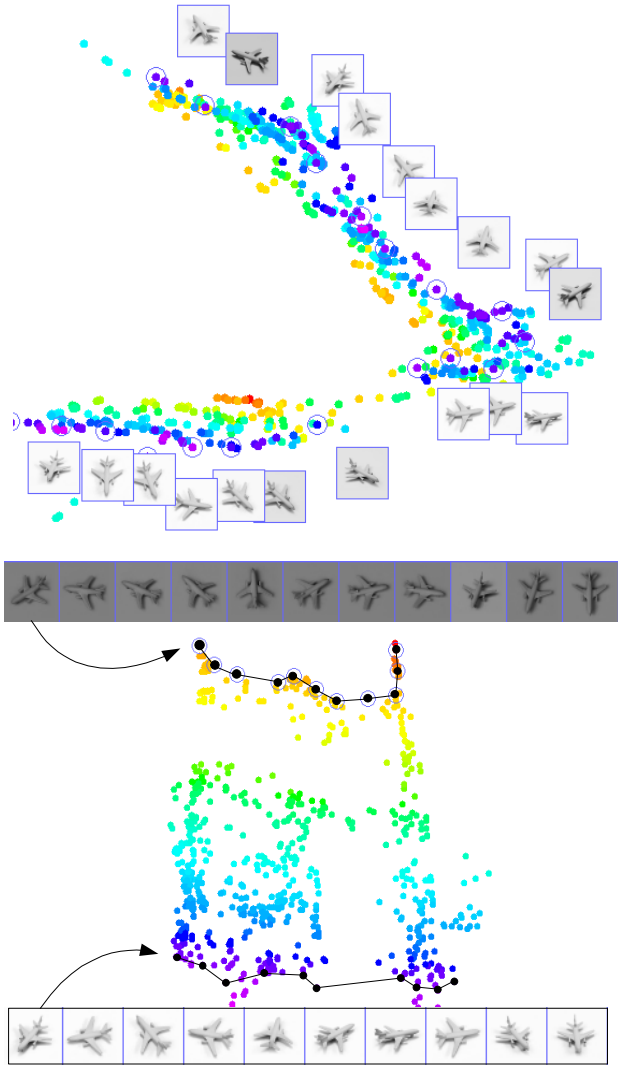


Figure 2.10: 3D embedding of NORB images by LLE algorithm. The neighborhood graph was constructed to create invariance to lighting, but the linear reconstruction weights of LLE force it organize the embedding by lighting. The shape of the embedding resembles a folded paper. The top image shows the 'v' shape of the fold and the lower image looks into the valley of the fold.

3
DrLIM Applied to Face Verification

Joint work with Sumit Chopra.

3.1 Introduction

Traditional approaches to classification using discriminative methods, such as neural networks or support vector machines, generally require that all the categories be known in advance. They also require that training examples be available for all the categories. Furthermore, these methods are intrinsically limited to a fairly small number of categories (on the order of 100). Those methods are unsuitable for applications where the number of categories is very large, where the number of samples per category is small, and where only a subset of the categories is known at the time of training. Such applications include face recognition and face verification: the number of categories can be in the hundreds or thousands, with only a few examples per category. A common approach to this kind of problem is distance-based methods, which consist in computing a similarity metric between the pattern to be classified or verified and a library of stored prototypes. Another common approach is to use probabilistic (generative) methods in a reduced-dimension space, where the model for one category can be trained without using examples from other categories. To apply discriminative learning techniques to this kind of application, we must devise a method that can extract information about the problem from the available data, without requiring specific information about the categories.

The solution presented here is to *learn a similarity metric from data* using the DrLIM approach. DrLIM learns a function that maps inputs to a low-dimensional manifold such that *similar* inputs are close in the output space. We demonstrated the approach in the previous chapter as way to learn invariance and discover underlying relationships in the data. In this chapter,

we demonstrate how DrLIM can be used to learn a similarity metric for classification. Unlike most classification approaches, in which instances of every class must be seen during training, a learned similarity metric can be used to compare or match new samples from previously-unseen categories (e.g. faces from people not seen during training). This method can be applied to classification problems where the number of categories is very large and/or where examples from all categories are not available at the time of training.

3.2 Face Verification with a Learned Similarity Metric

The task of face verification (Rizvi et al., 1998), is to accept or reject the claimed identity of a subject in an image. Performance is assessed using two measures: percentage of *false accepts* and the percentage of *false rejects*. A good system should minimize both measures simultaneously.

3.2.1 Previous Work

The idea of mapping face images to low-dimensional target spaces before comparison has a long history, starting with the PCA-based Eigenface method (Turk and Pentland, 1991) in which $G(X)$ is a linear projection trained non-discriminatively to maximize the variance. The LDA-based Fisherface method (Belhumeur et al., 1997) is also linear, but trained discriminatively so as to maximize the ratio of inter-class and intra-class variances. Non-linear extensions based on Kernel-PCA and Kernel-LDA have been discussed (Hsuan Yang et al., 2000). See (Shakhnarovich and Moghaddam, 2004) for a review of subspace methods for face recognition. One major shortcoming of all those approaches is that they are very sensitive to geometric transformations of the input images (shift, scaling, rotation) and to other variability (changes in facial expression, glasses, and obscuring scarves). Some authors have described similarity metrics that

are locally invariant to a set of known transformations. One example is the Tangent Distance method (Simard et al., 2000). Another example, which has been applied to face recognition, is elastic matching (Lades et al., 1993). Others have advocated warping-based normalization algorithms to maximally reduce the variations of appearance due to pose (Martinez, 2002). The invariance properties of all these models are hand-designed in advance. In the method described here, the invariance properties do not come from prior knowledge about the task, rather they are learned from data.

Our approach is to build a trainable system that constructs a non-linear mapping of images of faces to points in a low-dimensional space so that the distance between these points is small if the images belong to the same person and large otherwise. Learning the similarity metric is realized by training a network that consists of two identical convolutional networks that share the same set of weights - a *Siamese Architecture* (Bromley et al., 1993) (see Fig. 2.2). The loss functional that drives the learning is very similar to the DrLIM loss functional. As in DrLIM, the training set is composed of similar and dissimilar pairs. For the face verification task, we term similar pairs "genuine pairs" and term dissimilar pairs "impostor pairs". A pair of images is *genuine* if the subject is the same, and *impostor* if the subject is different.

3.3 Experiments

Three databases of face images were used for training, and testing was done on 2 of those databases. We will discuss the databases in detail and then explain the training protocol and architecture.

3.3.1 Datasets and Data Processing

The first round of training and testing was done with the AT&T Database of Faces (AT&T Faces Database,), consisting of 10 images each of 40 subjects, with variations in lighting, facial

Figure 3.1: Images from the AT&T dataset. The top row shows the variability in images of a single subject. The bottom row shows a genuine pair and an impostor pair.

expression, accessories, and head position. Each image is 112x92 pixels, gray scale, and closely cropped to include the face only (see Fig. 3.1.

There was no need to pre-process the images for size or lighting normalization, since one of the stated goals was to train an architecture that would be resilient to such variations. However, we did reduce the resolution of the images to 56x46 using 4x4 subsampling.

The second set of training and testing experiments was performed by combining two datasets: the AR Database of Faces, created at Purdue University and publicly available (Martinez and Benavente, 1998), and a subset of the grayscale Feret Database (FERET Faces Database,). Image pairs from both of these datasets were used in training, but only images from the AR dataset were used for testing.

The AR dataset comprises 3,536 images of 136 subjects with 26 images per subject. The images had a very high degree of variability in terms of expression, lighting, and artificial occlusions like sunglasses and face-obscuring scarves (see Fig. 3.2). A simple correlation-based centering algorithm was used to center the faces. The images were then cropped and reduced to

Figure 3.2: Images from the AR dataset. The top row shows the variability in images of a single subject. The bottom row shows a genuine pair and an impostor pair.

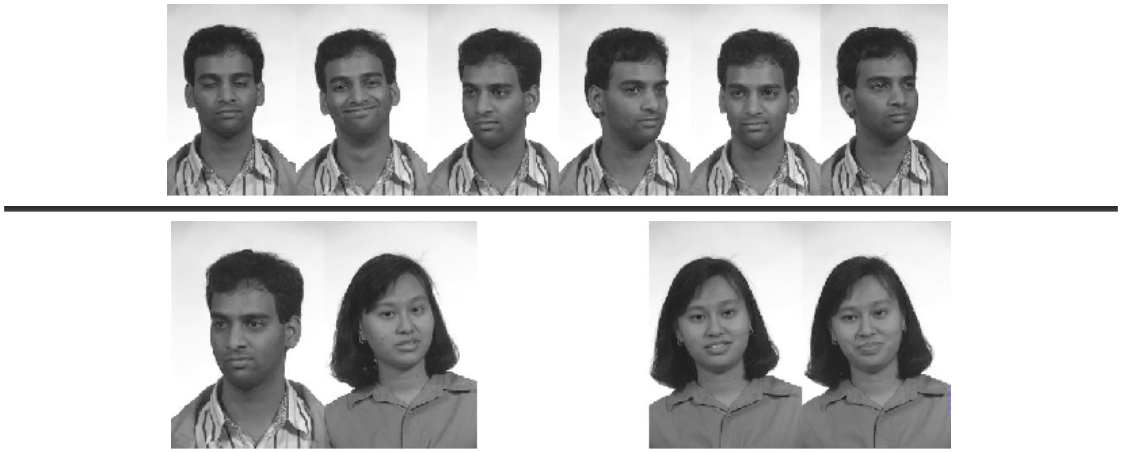


Figure 3.3: Images from the FERET dataset. The top row shows the variability in images of a single subject. The bottom row shows a genuine pair and an impostor pair.

56x46 pixels. Although the centering was sufficient for the purposes of cropping, there remained substantial variations in head position in many images.

The Feret Database, distributed by the National Institute of Standards and Technology, comprises 14,051 images collected from 1,209 subjects (see Fig. 3.3). We used a subset of the full database solely for training. The only preprocessing was cropping and subsampling to 56x46 pixels.

Partitioning For the purpose of testing using images not seen during training, the datasets were split into two disjoint sets, namely SET1 and SET2. Each image in each of these sets was paired up with every other image in that set to generate the maximum number of genuine pairs and impostor pairs.

For the AT&T data, SET1 consisted of 350 images of first 35 subjects and SET2 consisted of 50 images of last 5 subjects. Training was done using only the image pairs generated from SET1. Testing (verification) was done using the image pairs from SET2 and the unused image pairs from SET1.

For the AR/Feret data, SET1 contained all the Feret images and 2,496 images from 96 subjects in the AR database. SET2 contained the 1,040 images from the remaining 40 subjects in the AR database. The actual training set that was used contained 140,000 image pairs that were evenly split between genuine and impostor.

3.3.2 Training Protocol

Experiments using the AT&T dataset explored a number of different sub-net architectures. We only describe the best-performing architecture in the following sections. C_x denotes a convolutional layer, S_x denotes a sub-sampling layer, and F_x denotes a fully connected layer, where x is the layer index. The basic architecture is $C_1 - S_2 - C_3 - S_4 - C_5 - F_6$. Layer C_1 had 15

feature maps, of size 50x40, with kernel size of 7x7. The number of trainable parameters were 750. S_2 was a subsampling layer with kernels of size 2x2. Layer C_3 consisted of 45 feature maps, of size 20x15, and kernels of size 6x6 and 7128 trainable parameters. The feature maps of this layer were partially connected to those of S_2. The exact connections are in a pattern similar to (LeCun et al., 1998). The motivation is to break the symmetry, thereby pushing the feature maps to extract and learn different features. S_4 was a subsampling layer with kernels of size 5x5. Next was the C_5 layer with 250 feature maps and kernel size of 1x1. Finally there was a fully connected layer F_6 with 50 units. This was the output of the function G_W.

- C_1**.** Feature maps: 15; Size 50x40; Kernel size: 7x7. Trainable parameters: 750; Connections: 1500000.
  Fully Connected with the input.

- S_2**.** Feature maps: 15; Size: 25x20; Field of view: 2x2. Trainable parameters: 30; Connections: 37500.

- C_3**.** Feature maps: 45; Size: 20x15; Kernel size: 6x6. Trainable parameters: 7128; Connections: 2139600.
  Partially connected to S_2.

- S_4**.** Feature maps: 45; Size: 5x5; Field of view: 4x3. Trainable parameters: 100; Connections: 16250.

- C_5**.** Feature maps: 250; Size 1x1; Kernel size: 5x5. Trainable connections: 312750.
  Fully connected to S_4.

- F_6**.** Number of units: 50. Trainable parameters: 12550; Connections: 12550.

Figure 3.4: Internal state of the convolutional network for a particular example.

Training requires two sets of data: the training set, for actually learning the weights of the system, and the validation set, for testing the performance of the system during training. Periodical performance evaluation with the validation set allows us to control over-fitting.

Training the network was done with pairs of images taken from SET1. One half of the image pairs were genuine and one half were impostor, produced by randomly pairing images of different subjects. Periodical performance evaluation of the system was done using a validation set to avoid over-fitting. The validation set was composed of 1500 image pairs, taken from the unused pairs of SET1, and in the same 50% genuine, 50% impostor ratio as the training set.

The performance of the network was measured by a calculation of the percentage of impostor pairs accepted (FA), and the percentage of genuine pairs rejected (FR). This calculation was made by measuring the norm of the difference between the outputs of a pair, then picking a threshold value that sets a given trade-off between the FA and FR percentages.

	AT&T		AR/Purdue	
	Val	Test	Val	Test
Number of Subjects	35	5	96	40
Images/Subject	10	10	26	26
Images/Model	–	5	–	13
No. Genuine Images	500	500	750	500
No. Impostor Images	500	4500	750	4500

	False Accept		
	10%	7.5%	5%
AT&T (Test)	0.00	1.00	1.00
AT&T (Validation)	0.00	0.00	0.25
AR (Test)	11	14.6	19
AR (Validation)	0.53	0.53	0.80

Table 3.1: Above: Details of the validation and test sets for the two datasets. Below: False reject percentage for different false accept percentages.

3.4 Testing (Verification) and Results

Figure 3.4 shows the internal state of the convolutional network for a particular test image. The first layer extracts various types of local gradient features, as well as smooth features.

The system was tested for a face verification scenario. The system is given an image and asked to confirm the claimed identity of the subject in that image. We perform verification by comparing the test image with a Gaussian model of images of the claimed subject. The method is discussed below.

Testing was done on a test set of size 5000. It consisted of 500 genuine and 4500 impostor pairs. For the AT&T experiments the test images were from 5 subjects unseen in training. For AR/Feret experiments the test images were from 40 unseen subjects in the more difficult AR database.

The output from one of the subnets of the Siamese network is a feature vector of the input image of the subject. We assume that the feature vectors of each subject's image form a multivariate normal density. A model is constructed of each subject by calculating the mean and the variance-covariance matrix using the feature vectors generated from the first five images of each subject.

The likelihood that a test image is genuine, ρ_{genuine}, is found by evaluating the normal density of the test image on the model of the concerned subject. The likelihood of a test image being an impostor, ρ_{impostor}, is assumed to be a constant whose value is estimated by calculating the average ρ_{genuine} value of all the impostor images of the concerned subject. The probability that the given image is genuine is given by

$$\text{Prob(genuine)} = \frac{\rho_{\text{genuine}}}{\rho_{\text{genuine}} + \rho_{\text{impostor}}}$$

The verification rates obtained from testing the AT&T database and the AR/Purdue database

are strikingly different (see table 3.1), underlining the differences in difficulty in the two databases. The AT&T dataset is relatively small, and our system required only 5000 training samples to achieve very high performance on the test set. The AR/Purdue dataset is very large and diverse, with huge variations in expression, lighting, and added occlusions. Our higher error rates reflect this level of difficulty.

We have illustrated the DrLIM method with a face verification application. We chose to use a convolutional network architecture which exhibits robustness to geometric variations of the input, thereby reducing the need for accurate registration of the face images.

4

A Real-World Problem: Long-Range Vision on an Offroad Robot

Joint work with Ayse Erkan, Jan Ben, and Koray Kavukcuoglu.

4.1 Introduction

The problem of autonomous navigation has inspired decades of research, but the basic issues are very difficult and remain unsolved. If robotic vehicles could reliably and robustly drive through unknown terrain to a given location, the implications would be tremendous for exploration (both on Earth and extra-terrestrially), search and rescue operations, driving safety (due to the development of automatic obstacle avoidance systems), and other interests. Recent successes of autonomous vehicles have been well publicized. On Mars, two robotic rovers have been exploring and collecting data since 2004. The Mars rovers, however, are carefully monitored and controlled; they are not fully autonomous (Maimone et al., 2007; Matthies et al., 2007). Another prominent example was the 2005 DARPA Grand Challenge, which featured fully autonomous vehicles racing over a 132 mile desert course (Iagnemma and Buehler, 2006a; Iagnemma and Buehler, 2006b). However, the Grand Challenge required vehicles to drive autonomously from waypoint to waypoint along a desert road: an arguably easier task than offroad navigation through arbitrary terrain. We focus on the task of long-range vision on an autonomous mobile robot in offroad terrain.

Humans navigate effortlessly through most outdoor environments, detecting and planning around distant obstacles even in new, never-seen terrain. Shadows, hills, groundcover variation - none of these affect our ability to make strategic planning decisions based purely on visual

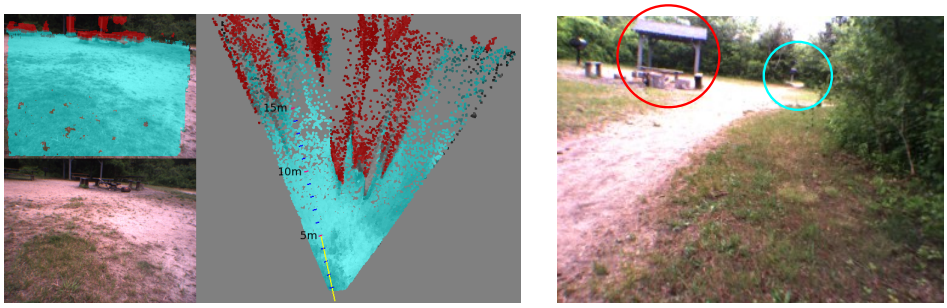


Figure 4.1: *Left:* Top view of a map generated from stereo (stereo is run at 320x240 resolution). The map is ”smeared out” and sparse at long range because range estimates from stereo become inaccurate above 10 to 12 meters.
Right: Examples of human ability to understand monocular images. The obstacles in the mid-range are obvious to a human, as is the distant pathway through the trees.

information. Human visual performance is not due to better stereo perception; in fact, humans are excellent at locating pathways and obstacles in monocular images (see Fig. 4.1 right).

Recent learning-based research has focused on increasing the range of vision by classifying terrain in the far field according to the color of nearby ground and obstacles. This type of near-to-far color-based classification is quite limited, however. Although it gives a larger range of vision, the classifier has low accuracy and can easily be fooled by shadows, monochromatic terrain, and complex obstacles or ground types.

The primary contribution of this work is a *long-range vision system that uses self-supervised learning to train a classifier in realtime*. To successfully learn in complex environments, the classifier must be trained with *discriminative features extracted from large image patches*, and the features must be labeled with *visually consistent* categories. For the classifier to successfully generalize from near to far field, the training samples must be *normalized with respect to scale and distance*. The first criterion, training with large image patches, is crucial for true recognition

of obstacles, paths, groundtypes, and other natural features. Color histograms or texture gradients cannot replace the contextual information in actual image patches. The second criterion, visually consistent labeling, is equally important for successful learning. The classifier is trained using labels generated by stereo processing. If the label categories are inconsistent or extremely noisy, the learning will fail. Therefore, our stereo-based supervisor module uses 5 categories that are visually distinct: *super-ground*, *ground*, *footline*, *obstacle*, and *super-obstacle*. The supervisor module is designed to limit incorrect or misaligned labels; multiple ground planes are estimated to threshold the stereo points, and false obstacles are removed through analysis of plane distance statistics. The third criterion, normalization with respect to size and distance, is necessary for good generalization. We normalize the image by constructing a horizon-leveled input pyramid in which similar obstacles are a similar height, regardless of their distance from the camera.

The long-range vision classifier was developed and tested as part of a full navigation system. The outputs from the classifier populate a hyperbolic polar coordinate costmap, and planning algorithms are run on the map to decide trajectories and wheel commands at each step. Since the classifier is trained online, its outputs vary over time as the inputs and training labels change. To accommodate this uncertainty, we use histograms to accumulate the classifier probabilities over time. Histograms allow us to accumulate evidence for a particular label in the face of changing classifier outputs. Similarly, the geometry of the hyperbolic polar map is designed to accommodate the range uncertainties that are intrinsic to image-space obstacle labeling, while also providing a vehicle-centered world representation with an infinite radius using a finite number of cells. The mapping and planning approach is discussed in Chapter 5

4.1.1 The LAGR Program and Platform

The vision system described here was developed on the LAGR platform (see Fig. 4.3). LAGR (Learning Applied to Ground Robots) is a DARPA program that ran from 2005 to 2008 with 10 research participants. Unlike the DARPA Grand Challenge, in which competitors could build their own platform with any autonomous sensors or processing power, the LAGR program provided fully equipped robotic platforms to its competitors and prohibited them from modifying the hardware in any way. This forced participants to focus on development of learning and vision algorithms for the vehicle, which included only passive stereo vision. LAGR was also distinguished by its rigorous testing regime. With a baseline navigation system developed by CMU as a comparison metric, the participants were tested monthly by a DARPA testing team and required to beat the baseline system by a factor of 2 by the midpoint of the program. The testing followed a strict protocol. Each month, the LAGR assessment team set up a course in unknown terrain, traveling to such disparate locales as San Antonio, TX, Washington, D.C., and Vermont (examples of terrain from 8 DARPA tests are shown in Figure 4.2). The navigation software provided by each team was loaded in turn onto a LAGR robot, and the software was started. The goal, designated by GPS, was up to 200 meters away and generally out of sight from the start point. Each team was run 3 times in succession, and a composite score was given that took into account the total time to reach the goal as well as the final distance from the goal (if the robot did not complete the course).

The LAGR robot (see Fig. 4.3) has 4 onboard computers, 2 stereo camera pairs with a maximum resolution of 1024x768, a GPS receiver, and an IMU (inertia measurement unit). The 4 onboard computers (one for low-level motor control, one for planning, and one for each "eye") are 2Gz processors with 1GB memory, connected with a 1Gb ethernet. The vehicle is approximately 1 meter long and 1 meter tall and has a maximum speed of 1.2 meters/second. The

Figure 4.2: Images from 8 different official LAGR test courses.

hardware and the baseline navigation software were developed by Carnegie Mellon University and the National Robotics Engineering Center (NREC). The platform and the program have been thoroughly documented in previous publications (Jackel, 2005; Jackel et al., 2006).

4.2 Learning Traversability of Long-Range Visual Data

The problem is to predict the traversability of terrain from visual inputs. The traversability of nearby areas can be determined by running a stereo algorithm and using various heuristics on the resulting 3d point cloud, resulting in a labeled training set at time t, containing samples for

Figure 4.3: The LAGR mobile robotic vehicle, developed by Carnegie Mellon University's National Robotics Engineering Center. Its sensors consist of 2 stereo camera pairs, a GPS receiver, and a front bumper.

nearby areas only, within a range d:

$$\mathcal{S}_t = \{(X^i, y^i) \mid \text{dist}(i, \text{camera}) < d\}.$$

The training samples X^i are image data: input windows extracted from the full camera image. The correspondences between these images and the labels from stereo need to be very robust. There are two options for establishing correspondences: first, the windows could be projected onto the estimated geometry of the scene, thus giving robot-relative coordinates on the ground (no z information). Then the stereo algorithm could be run on grid cells of points on the ground and the same coordinates could be used. Second, the stereo heuristics could be estimated in the image space, establishing a direct correspondence with the visual windows with no projection necessary. This is the more robust, and more desirable, option.

Given the labeled image data, plus the camera's position at time t (pose has 6 degrees of freedom: $\text{pose}_t = [x, y, z, \theta_{roll}, \theta_{pitch}, \theta_{yaw}]$), the task is to predict the traversability of the

entire image, up to a range D:

$$\mathcal{T}_t = \{(X^i) \mid \text{dist}(i, \text{camera}) < D\}.$$

The proposed solution involves learning 2 parametric functions, G_{W1} and H_{W2}, such that their composition produces a mapping from RGB image patch to traversability cost. The function G_{W1}, a feature extractor is trained offline to produce features that are not specialized to a single environment and are invariant to displacement, scale, and other transformations. The function H_{W2}, a supervised classifier, is trained in realtime on the training set $\mathcal{S}_t$. H_{W2} has weight decay towards a static set of parameters $W0$ that are trained over multiple terrain types.

The architecture of the long-range vision module incorporates G_{W1} and H_{W2} in a framework that processes input images and produces traversability costs. On each input frame, a full cycle of feature extraction, training, and classification is executed.

4.2.1 Overview of Navigation and Long-Range Vision

The LAGR platform is accompanied by navigation software developed by NREC and Carnegie Mellon. The navigation system uses a stereo-based obstacle detection module to populate a Cartesian coordinate map on every frame. The *D-star* algorithm is run on the global cost map to plan paths and generate driving commands. Our navigation system makes no use of this baseline software. Instead, our system uses multiple levels of perception and planning. At the lowest level, perception is simple and short-range, using very low resolution images. This allows very fast response times and robust obstacle avoidance, even allowing the vehicle to dodge moving obstacles. At the highest level, the perception and planning processes are much more sophisticated and use higher resolution images (see Table 4.1). The frequency and latency of these processes is much slower, but, because they are responsible for long-range vision and planning, a slower response time is acceptable. This multi-resolution architecture has been described in

Control loop	Resolution	Range	Map type	Perception
Fast (10Hz)	180x160	0 to 5 m	Cart. grid (10 cm)	stereo
Slow (2-3Hz)	512x384	4 to 12 m	Cart. grid (20 cm)	stereo
Slow (2-3Hz)	512x384	5 to ∞ m	hpolar (.2 - 25 m)	classifier

Table 4.1: The table shows the 3 levels of perception, mapping and planning. The fast, short-range stereo is run on a separate process from the other two perception mechanisms.

other publications (Sermanet et al., 2007).

The long-range vision system is a self-supervised, realtime learning process (see Fig. 4.4). It continuously receives images, generates supervisory labels, trains a classifier, and classifies the long-range portion of the images, completing one full training and classification cycle every half second. The only inputs are a pair of stereo-aligned images and the current position of the vehicle, and the output is a set of points in vehicle-relative coordinates where each point is labeled with a vector of 5 probabilities that correspond to 5 possible categories. The points and their energy vectors are used to populate a vehicle-relative polar-coordinate map which combines constant radius cells for the first 15 meters and hyperbolically increasing radius cells for cells from 15 meters to infinity. Path planning algorithms are run on the polar map, producing path candidates which interact with the short-range obstacle avoidance module to produce driving commands. The primary components of the learning process are briefly listed.

- **Pre-processing and Normalization.** Pre-processing is done to level the horizon and to normalize the height of objects such that their pixel height is independent of their distance from the camera.

- **Stereo Supervisor Module.** The stereo supervisor assigns class labels to close-range windows in the normalized input. This is a complicated process involving multiple ground

plane estimation, heuristics, and statistical false obstacle filtering in order to generate training labels with as little noise as possible.

- **Feature Extraction.** Features are extracted from input windows in order to reduce dimensionality and gain a more discriminative, concise representation. Experiments are run with several different feature representations. The filters are trained offline.

- **Training and Classification.** The classifier is trained on every frame for fast adaptability. We use stochastic gradient descent and a cross entropy loss function.

- **Costmap Accumulation.** The classifier outputs are accumulated in a hyperbolic polar map using histograms. Likelihood vectors from each point are added into histograms in appropriate cells in the map, the histograms are smoothed, and finally the histograms are mapped to traversability costs.

4.3 Horizon-Leveling and Normalization

We are strongly motivated to use large image patches (large enough to fully capture a natural scene element such as a tree or path) because larger context and greater information yields better learning and recognition. However, the problem of generalizing from nearby objects to far objects is daunting, since apparent (pixel) size scales inversely with distance: Pixel size $\propto \frac{1}{\text{Distance}}$. Our solution is to create a normalized "pyramid" of 7 sub-images which are extracted at geometrically progressing distances from the camera. Each sub-image is subsampled according to its estimated distance, yielding a set of images in which similar objects have a similar pixel height, regardless of their distance from the vehicle (see Fig. 4.5). The closest pyramid row has a target range of 4 to 11 meters distance and is subsampled with a scaling factor of 6.7. The furthest pyramid row has a range from 112 meters to infinity (above the horizon) and has a

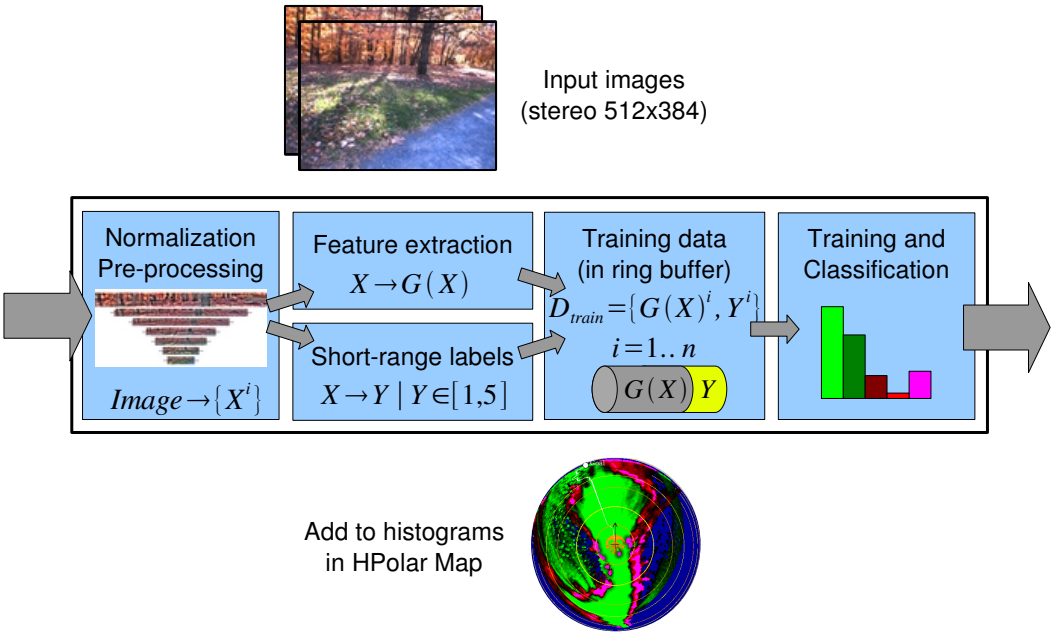


Figure 4.4: The input to the long-range vision module is a pair of stereo-aligned images and the current position and bearing of the robot. The images are normalized and features and labels are extracted, then the classifier is trained and immediately used to classify the entire image. The classifier outputs are accumulated in histograms in a hyperbolic polar map according to their vehicle-relative coordinates.

Figure 4.5: The input image at left has been systematically cropped, leveled and subsampled to yield each pyramid row seen to the right. The bounding boxes demonstrate the effectiveness of the normalization: trees that are different scales in the input image are similarly scaled in the pyramid.

scaling ratio of 1 (no subsampling). The image row number to distance correlation is obtained by estimating the ground plane position; this estimate is calculated on each frame through a process described in Section 4.4.2.

A bias in the roll of the cameras, plus the natural bumps and grading in the terrain, means that the horizon is usually skewed in the input image. We normalize the horizon position in the pyramid by explicitly estimating the horizon's location and then warping the image in relation to it. First we estimate the groundplane $P = (a, b, c, d)$ using a Hough transform on the stereo point cloud, then refine that estimate using a PCA robust refit (see Section 4.4.2 for details). P is converted to (p_r, p_c, p_d, p_o) format (p_r = *row*, p_c = *column*, p_d = *disparity*, p_o = *offset*), and the horizon can be leveled by computing the four corners of the target subimage (points A, B, C,and D in Fig. 4.6) and transforming that sub-image to a scaled target rectangle using an affine warp. For a row in the pyramid, we first compute the location of $\overline{\text{EF}}$ that lies on the ground plane at a distance with stereo disparity of d pixels. The endpoints of $\overline{\text{EF}}$ are found by computing the center of the line (M) by plane intersection, then finding the endpoints (E,F) by θ rotation:

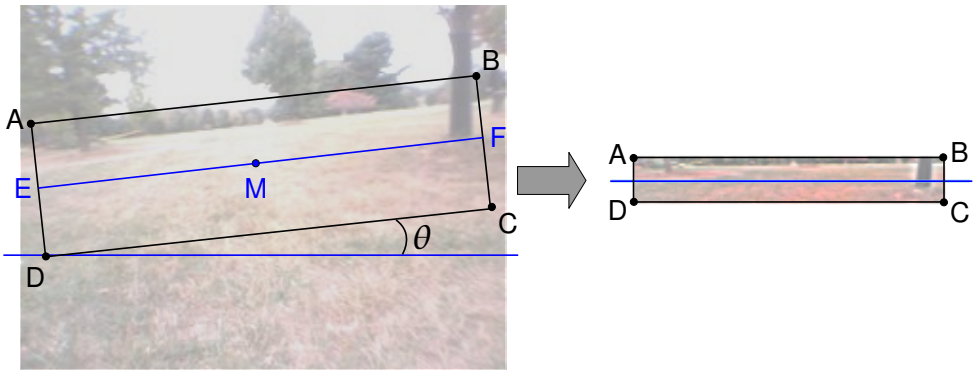


Figure 4.6: Each row in the normalized, horizon-leveled pyramid is created by identifying the 4 corners of the target sub-image, which must be aligned with the ground plane and scaled according to the target distance, and then warping to a re-sized rectangular region.

$$\mathrm{M}_y = \frac{p_c * \mathrm{M}_x + p_d * d + p_o}{-p_r}$$

$$\mathrm{E} = (\mathrm{M}_x - \mathrm{M}_x \cos\theta \ , \ \mathrm{M}_y - \mathrm{M}_x \sin\theta)$$

$$\mathrm{F} = (\mathrm{M}_x + \mathrm{M}_x \cos\theta \ , \ \mathrm{M}_y + \mathrm{M}_x \sin\theta)$$

where M_x is the horizontal center of the image, and θ is found by projecting the left and right columns of the image:

$$\theta = \left(\frac{\mathrm{W} * p_c + p_d + p_o}{-p_r} - \frac{0 * p_c + p_d + p_o}{-p_r} \right) / \mathrm{W},$$

where W is the width of the input image. Points A, B, C, and D can be found by rotating and scaling E and F by the scaling value (α) for the pyramid row:

$$\mathrm{A} = (\mathrm{E}_x + \alpha \sin\theta, \mathrm{E}_y - \alpha \cos\theta)$$

$$\mathrm{B} = (\mathrm{F}_x + \alpha \sin\theta, \mathrm{F}_y - \alpha \cos\theta)$$

$$\mathrm{C} = (\mathrm{F}_x - \alpha \sin\theta, \mathrm{F}_y + \alpha \cos\theta)$$

$$\mathrm{D} = (\mathrm{E}_x - \alpha \sin\theta, \mathrm{E}_y + \alpha \cos\theta)$$

The input is also converted from RGB to YUV, and the Y (luminance) channel is contrast normalized to alleviate the effects of hard shadow and saturation. The contrast normalization performs a smooth neighborhood normalization on each y in Y by normalizing by the linear sum of a smooth 16x16 kernel and a 16x16 neighborhood of Y (centered on y). Pixel x in image I is normalized by the values in a soft window centered on x:

$$x = \frac{x}{\sum_{y \in I^W, k \in K} yk + 1}$$

where I^W is a 16x16 window in I, and K is a smooth, normalized 16x16 kernel.

4.4 Realtime Stereo Supervision

The supervision that the long-range classifier receives from the stereo module is critically important. The realtime training can be dramatically altered if the data and labels are changed in small ways, or if the labeling becomes noisy. Therefore, the goal of the supervisor module is to provide data samples and labels that are *visually consistent*, *error-free*, and *well-distributed*. The basic approach begins with a disparity point cloud generated by a stereo algorithm, and has 4 steps:

1. A **Stereo Algorithm** produces a 3D point cloud.

2. **Estimation of the ground plane** within the point cloud is done, allowing separation of the points into ground and obstacle classes.

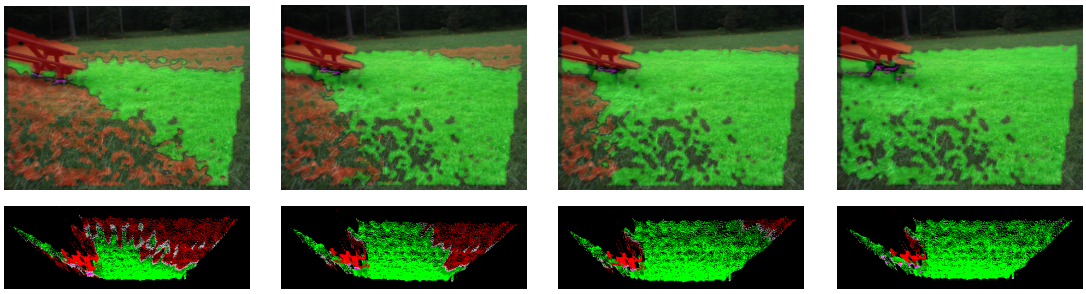
3. **Projection** Next, the obstacle points are projected onto the ground plane to locate the feet of obstacles.

4. **Labeling**. Third, overlapping regions of points are considered and heuristics are used to assign each region to one of five categories.

The results from this basic method have several sources of error, however. Since offroad terrain is rarely perfectly flat, areas of traversable ground can stick up above the ground plane and be misclassified as obstacle. Also, tufts of grass or leaves can look like obstacles when a simple plane distance threshold is used to detect obstacles. These sorts of error are potentially disastrous to the classifier, so two strategies, multi-groundplane estimation and moments filtering, are employed to avoid them. This section describes the stereo algorithm and the supervision process in detail.

4.4.1 The Stereo Algorithm: Triclops

The LAGR vehicle is equipped with 2 sets of Bumblebee stereo cameras, one for the left field and one for the right field, which together comprise a field of view of 120°. The Triclops SDK, from Point Grey Research, provides image rectification and stereo processing on each pair of cameras (Point Grey Research Inc., 2003). Rectification is done to remove lens distortions and misalignments. Stereo processing produces a range estimate for each valid pixel in the field of view, and is limited by the resolution of the input images; our system uses 384x512 images, which provides a range of 12 to 15 meters with the Triclops algorithm. The Triclops algorithm works by triangulating between 2 slightly offset cameras: correspondences between pixels in the two images are found and distances are calculated based on the geometry of the cameras. The correlation is made with a sum of absolute differences over band-passed images. The algorithm fails if sufficient texture and contrast is not present in the images, and can be fooled by repeating visual elements (fenceposts, tall grass) and visual artifacts (sun glare, specularities).



a. 1 ground plane b. 2 ground planes c. 3 ground planes d. 3 planes + filtering

Figure 4.7: Up to 3 ground planes are found in the stereo point cloud of input images. After a plane is found, points close to the plane are removed from the point cloud and the process is repeated (a, b, c). After ground plane estimation, statistical analysis of the plane distances of the points is done to filter out remaining false obstacles (d).

4.4.2 Ground Plane Estimation

It is necessary to locate the ground plane in order to distinguish ground from obstacle points within a stereo point cloud. However, the assumption that there is a single perfect ground plane is rarely correct in natural terrain. To relax this assumption, we find multiple planes in each input image and use their combined information to divide the points into ground and obstacle clouds. First we describe how to fit a single ground plane model to a point cloud, where the point cloud is a set of 3-tuples: $\mathcal{S} = \{(x^i, y^i, z^i) \mid i = 1..n\}$ where x^i, y^i, z^i defines the position of the point relative to the robot's center. Color components (r^i, g^i, b^i) and image relative coordinates $(\text{row}^i, \text{column}^i, \text{disparity}^i)$ are also associated with each point in $\mathcal{S}$.

Ground plane estimation is done in two stages: an initial estimate is found using a Hough transform, and principal component analysis is used to refine the estimate. A Hough transform (Duda and Hart, 1972) is a voting procedure that is used to select a shape from within a parameterized class of shapes (in this case, the class of planes). A quantized parameter space

defines the set of possible planes, and points in the cloud "vote" for a single plane. The winning plane is defined by the parameter vector that accumulates the most votes. The ground plane parameter space is constrained in pitch, roll, and offset, and quantized into 64 bins for each parameter. The bounding of each parameter was governed by consideration of the maximum slope on which the robot could reasonably drive.

The exact voting process begins by randomly selecting a subset of points from $\mathcal{S}$. As the algorithm iterates each of these points, a point (x^i, y^i, z^i) in $\mathcal{S}$ votes for each of the candidate planes that it intersects. This can be done efficiently: we precompute all possible normals $[abc]$, then increment the tally for the candidate with parameters $a, b, c, (x^i a + y^i b + z^i c)$ if all the parameters are within the defined bounds. The yaw parameter c is fixed at 1.0, so the space exploration is only in the 3 dimensions of pitch, roll and offset. After the voting process is done, a plane is selected by finding the maximum within the voting space:

$$X = P_{ijk} \ \mid i, j, k = \text{argmax}_{i,j,k} \left(V_{ijk} \right)$$

where X is the new plane estimate, V is a tensor that accumulates the votes, and P is a tensor that records the plane parameter space.

The Hough estimate of the ground plane is then refit using principal component analysis. The PCA refit begins with the set of *inlier* points that are within a threshold of the Hough plane, then computes the eigenvalue decomposition of the covariance matrix of the points $X^{1..n}$:

$$\frac{1}{n} \sum_{1}^{n} X^i X^{i\prime} = Q \Lambda Q'$$

The eigenvector in Q corresponding to the smallest eigenvalue in Λ is the new plane normal, and the offset of the plane is set to the mean offset of the inlier point set.

The process of plane fitting that has been described is fast and robust, fitting a correct plane even in challenging terrain such as narrow paths where very few points on the ground can be

seen. Restricting the Hough transform parameter space is necessary to prevent the plane being fit to table tops or vertical planes of buildings or cars. However, as has already been mentioned, natural terrain is often not planar. It can be non-continuous or disjoint (a street and a sidewalk, for instance) or curving (undulations of the ground, hills, and valleys). Since fitting a single plane to a non-planar surface can produce disastrous results (a hill or a valley will appear to be an obstacle if the points are far enough from the estimated plane) both for driving and for training the classifier, we fit multiple planes to each frame in the hopes of capturing the dominant facets of the terrain. The multi-plane fitting process is iterative in nature: after the first ground plane is fit to the point cloud, all points that are within a tight threshold of the plane are removed and a new plane is fit to the remaining points. The process continues until a stopping criterion is met: either (1) no valid plane can be found, (2) there are not enough points left in the point cloud, or (3) a maximum of 3 planes have been fit to the data.

More formally, we distinguish the set of "ground" points as a subset of the full point cloud: $\mathcal{S}^G \subseteq \mathcal{S}$, where $\mathcal{S}^G$ denotes the set of ground points. Given a set of m planes, $\mathcal{P} = \{P^i \mid i = 1..m\}$, points can be assigned to $\mathcal{S}^G$ based on their normal distance from the planes and a threshold α:

$$\mathcal{S}^G = \{(x^i, y^i, z^i) \mid D(X^i, P^j) \leq \alpha\}$$

where D computes the euclidean distance between point X^i and its projection on plane P^j:

$$D(X^i, P^j) = |x^i a^j + y^i b^j + z^i c^j + d^j|$$

Figure 4.4.2 shows the results of multiple plane fitting on a single point cloud. Frame *1* shows the point cloud in image space (top) and in 3d (bottom) with a single plane fit to the scene. Frame *2* shows the scene with a second plane fit, and Frame *3* shows the scene with 3 planes. The fourth panel shows the final classification of the points, after filtering by moments has been done. We discuss moments filtering now.

Even multiple ground planes cannot remove all error from the stereo labeling process. Tufts of grass or disparity mismatches (common with repeating textures such as tall grass, dry brush, or fences) can create false obstacles that cause poor driving and training. Therefore, our strategy is to consider the first and second moments of the plane distances of points and use the statistics to reject false obstacles. We use the following heuristics: if the mean plane distance is not too high and the variance of the plane distance is very low, then the region is traversable (probably a traversable hillside). Conversely, if the mean plane distance is very low but the variance is higher, then that region is traversable (possibly tall grass). These heuristics are simple, but they reduce errors in the training data effectively. The process is described in more detail.

The *plane distance* of each point in $\mathcal{S}$ is computed by projecting the point onto each plane and recording the minimum distance: $pd(X^i, \mathcal{P}) = min_{P \in \mathcal{P}}(x^i a + y^i b + z^i c + d)$, where $X^i = (x^i, y^i, z^i)$ is a point in $\mathcal{S}$ and $P = (a, b, c, d)$ defines a plane in $\mathcal{P}$. We next compute the mean and variance of non-overlapping regions of point plane distances, where the regions are defined in image space. Since each point is associated with its *(row, column)* indices, finding the points that correspond to a particular region is straightforward. For a region A centered on (r, c) (dimension was 10x10 pixels), the mean and variance follow:

$$\mu = \overline{A_{rc}} = \frac{1}{|A_{rc}|} \sum_{X^i \in A_{rc}} pd(X^i, \mathcal{P})$$

$$\nu = \text{Var}(A_{rc}) = \frac{1}{|A_{rc}|} \sum_{X^i \in A_{rc}} [pd(X^i, \mathcal{P}) - \mu]^2.$$

Proceeding according the heuristics described previously, a low and high threshold was established for both mean and variance and each region was analyzed. If ($\mu <$ low-mean $\wedge$ $\nu <$ high-var) or ($\mu <$ high-mean $\wedge$ $\nu <$ low-var), then the points within that region are moved to the traversable point cloud ($\mathcal{S}^G$). This false-obstacle filtering reduces the noise in the stereo supervision substantially. It is useful when the curvature of the ground exceeds the modeling capacity of the 3 ground planes, or when tufts of grass or leaves poke above an otherwise flat plane.

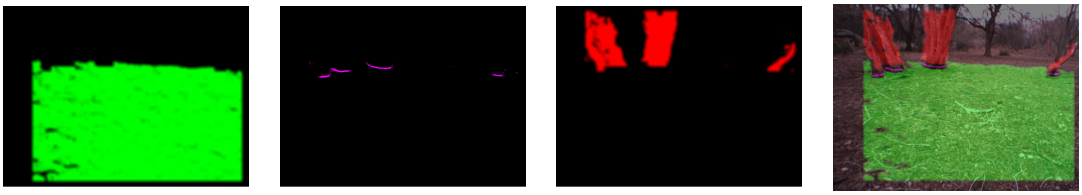
Frame 4 of Figure 4.4.2 shows the effect of moments filtering on a single frame of difficult, hilly terrain.

4.4.3 Footline Projection

Identifying the footlines of obstacles is critical for the success of the long-range vision classifier. Footlines are not only visually distinctive and thus relatively easy to model, they are also, by definition, at ground level, and thus we have more confidence in their exact location when they are mapped into 3d coordinates. There is a fundamental uncertainty about the exact distance of points that are beyond the range of stereo, and points that belong to obstacles are especially uncertain, because their height above the ground gets projected into false distance. Footlines have less uncertainty, because we know that a footline point has a height of 0. Footlines also define the border between traversable and non-traversable regions, so they are very significant for planning purposes.

Accordingly, we are not satisfied with robustly separating ground and obstacle point clouds ($\mathcal{S}^G$ and $\mathcal{S} - \mathcal{S}^G$). We find footlines by projecting obstacle points onto the ground planes. Concentrations of projected points are recognized as footlines. Any roughly vertical obstacle will project enough points onto the ground to be recognized as a footline. A gradually rising obstacle, however, such as a gentle transition from grass to scrub to bushes to trees, will probably not project enough points to form a footline. This is acceptable; often gradually transitioning terrain does not have a visually identifiable footline and thus makes poor training examples. For these areas, we can rely on our recognition of obstacles and forgo footline classification. To find footline points, each point in $\mathcal{S} - \mathcal{S}^G$ is projected onto the nearest ground plane in $\mathcal{P}$ and its *(row, column)* image space coordinates are recorded.

After ground plane estimation and footline projection, we are left with 3 sets of points: ground, obstacle, and footline. Each of these sets is mapped into the image plane, yielding 3



Ground map (**G**) Footline map (**F**) Obstacle map (**O**) Labeled image

Figure 4.8: The correct labeling of an image is dependent on robustly separating the stereo points into 3 subsets: ground points, footline points, and obstacle points. After these subsets are identified, final labels can be assigned, yielding the 5-category labeled image.

labeled "maps", which we denote **G** (ground-map), **F** (footline-map), and **O** (obstacle-map) (see Fig. 4.8). The final step toward stereo supervision considers the distribution of points within each overlapping window in the 3 maps, where the kernel size is the same as the input window size of the classifier. In order to take the context of the window's full field of view into consideration while emphasizing the content at the center of the window, weighted averages of the points in **G**, **F**, and **O** are computed, where the averages are peaked at the center of the window. This is efficiently done by convolving a separable Gaussian kernel over the 3 maps. Given the weighted average of ground, obstacle, and footline points present in a window whose center is at (i, j), a series of heuristics, summarized in Table 4.2, is applied that assigns a label to that window.

4.4.4 Visual Categories

Most classifiers that attempt to learn terrain traversability are binary; they only learn ground vs. obstacle. However, our classifier uses 5 categories: *super-ground*, *ground*, *footline*, *obstacle*, and *super-obstacle*. *Super-ground* and *super-obstacle* refer to input windows in which *only* ground or obstacle points are seen, and our confidence is very high that these labels are correct. The weaker *ground* and *obstacle* categories are used when mixed types of points are seen in the window, or

Category	Criteria for Label

Super-Traversable	*high number of ground points, very few obstacle and footline points.*
	if $(\mathbf{G}_{ij} > 0.8)$ **and** $(\mathbf{O}_{ij} < 0.1)$ **and** $(\mathbf{F}_{ij} < 0.01)$ **then**
	$Y_1 = 1$
	end if

Traversable	*some ground points, few footline points.*
	if $(\mathbf{G}_{ij} > 0.4)$ **and** $(\mathbf{F}_{ij} < 0.1)$ **then**
	$Y_2 = \mathbf{G}_{ij}$
	end if

Footline	*high number of footline points, some ground points.*
	if $(\mathbf{F}_{ij} > 0.6)$ **and** $(\mathbf{G}_{ij} > 0.1)$ **then**
	$Y_3 = 1$
	end if

Obstacle	*some obstacle points, few footline points.*
	if $(\mathbf{O}_{ij} > 0.4)$ **and** $(\mathbf{F}_{ij} < 0.1)$ **then**
	$Y_4 = \mathbf{O}_{ij}$
	end if

Super-Obstacle	*high number of obstacle points, very few ground and footline points.*
	if $(\mathbf{O}_{ij} > 0.8)$ **and** $(\mathbf{G}_{ij} < 0.1)$ **and** $(\mathbf{F}_{ij} < 0.01)$ **then**
	$Y_5 = 1$
	end if

Table 4.2: The rules for assigning a label to a window centered at (i, j) are given, based on the weighted averages in three point maps: $\mathbf{G}$ (ground-map), $\mathbf{F}$ (footline-map), and $\mathbf{O}$ (obstacle-map).

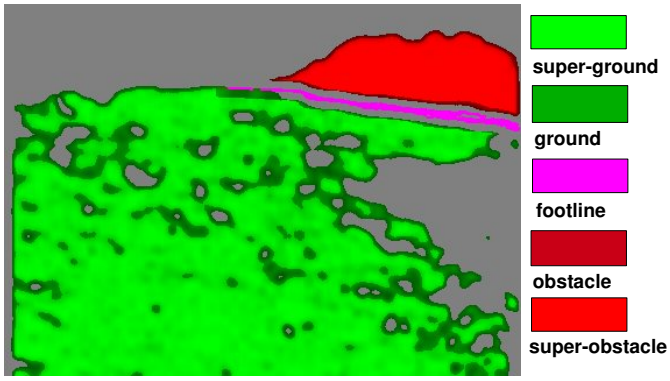


Figure 4.9: This shows the 5 category labeling of a full image.

when the confidence is lower that the label is correct. *Footline* is the label for input windows that have the footline points centered in the window. Obstacle feet are visually distinctive, and it is important to put these samples in a distinct category. Figure 4.9 and Figure 4.10 show examples of the 5 categories. Although the examples in each of these categories are still very diverse - pavement, leaves, and hard shadows are all found in the ground class, and trees trunks, buildings, and leafy bushes are all found in the obstacle class - they are more consistent than the broad categories of a binary classifier. Also, as mentioned before, obstacle footlines are a very important category because we can have higher confidence about their projection into Cartesian coordinates. However, if a binary classification scheme is used, then obstacle footlines must be interpreted as the "unknown" output of the classifier, since the footline necessarily inhabits the threshold between the 2 categories of a binary classifier.

4.5 Feature Learning

Normalized overlapping windows (3 channels, 25x12 pixels) from the pyramid rows provide a basis for strong near-to-far learning, but the high dimensionality makes it infeasible to directly

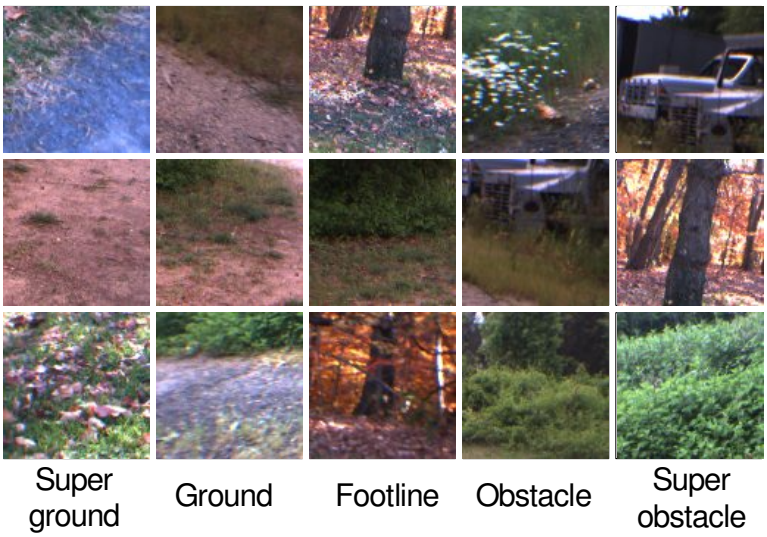


Figure 4.10: Examples of the 5 categories. Although the classes include very diverse training examples, there is still benefit to using more than 2 classes. *Super-ground*: only ground is seen in the window; high confidence, *ground*: ground and obstacle may be seen in window; lower confidence, *footline*: obstacle foot is centered in window, *obstacle*: obstacle is seen but does not fill window; lower confidence, and *super-obstacle*: obstacle fills window; high confidence.

train a classifier on the YUV windows. Feature extraction lowers the dimensionality while increasing the generalization potential of the classifier by incorporating invariance to irrelevant transformations in the input. The range of possible feature representations is enormous, only sparsely covered by the survey given in Section 1.2. We focus on *learned* representations with the belief that learning from real world data is preferable to hand-tuned features. Color histograms are fast to compute, but they are too limited. From a practical viewpoint, learned features are fast to compute, lower dimensional, and can be densely computed across an image. Learned features are efficient and precise, because they can capture patterns directly from the data. A trained feature representation can encode all the information in the input if it is trained to have a low reconstruction error, and a supervised learning algorithm can result in a highly discriminative feature representation.

Over the course of the LAGR program, many different feature representations were designed, trained, and compared. The progression of approaches is revealing of our changing understanding of appropriate feature representations for vision in natural environments. The first method implemented was a Radial Basis Function layer, with RBF centers initialized with k-means clustering over training images. Next, a convolutional network was trained with close-range labeled data collected from the stereo supervisor. Looking for a more general learned feature representation, we next trained an auto-encoder with a reconstruction criterion, and applied the filters convolutionally as a feature extractor. This approach was improved on with supervised fine-tuning. As a last avenue of investigation, co-location labeled data was collected and a convolutional net was trained with DrLIM. Similarly to the auto-encoder experiments, the DrLIM filters were then fine-tuned with supervised training on the labeled dataset. The compiled results of this progression of experiments are given in this section and in Chapter 6. Because of the realtime constraints of the navigation system, each method is computationally equivalent, as compared in the number of multiply-add operations and also as bounded by the same processing time.

4.5.1 Feature Extractor Training Procedures and Datasets

The datasets for training the feature extractors consist of samples, either labeled or unlabeled, taken randomly from 130 diverse logfiles. Image preprocessing and category labeling of these samples was identical to the online process described previously (see Sections 4.3 and 4.4). The datasets were compiled offline, and all training of the feature extractors was done offline. The trained filters were then copied to the robot and used in realtime for feature extraction, with no additional training or modification. Three datasets were collected from the same set of logfiles.

Unlabeled Dataset

A set of 500,000 unlabeled patches was randomly cropped from the logfiles. This data was used to train the k-means/RBF approach and the auto-encoder.

Five Class Labeled Dataset

A set of 500,000 labeled patches (balanced over five classes) was collected from the same logfiles by running the stereo supervisor on each frame and sampling from the labeled (close-range) windows. This labeled dataset was used to train the supervised convolutional net and to "fine-tune" the auto-encoder and DrLIM filters. The dataset was partitioned into training and test sets with 450,000 and 50,000 samples respectively.

Co-Location Dataset (Weakly Labeled)

A set of approximately 1.8 million unlabeled patches was collected, and approximately 2.6 million co-located pairs within this set were matched and recorded. This dataset was used to train the DrLIM convolutional net. The co-location information was calculated by finding correspondences efficiently with a spatially-indexed quad-tree. Each patch is entered into a quad-tree, referenced by its 2d world coordinates. Subsequent patches query the quadtree to locate patches that are within a narrow radius of the querying patch. Co-located pairs vary in viewpoint, scale,

Figure 4.11: The 100 radial basis function centers used for feature extraction. The centers were learned through unsupervised k-means clustering on a set of 500,000 diverse patches from log files.

occlusions, translation, and lighting.

4.5.2 Radial Basis Functions

Our first approach for feature learning is to learn a set of radial basis functions from data. For each 10x10 YUV input window in the normalized image pyramid, a feature vector is constructed by taking Euclidean distances between the input window and each of 100 fixed RBF centers. For an input window X and a set of n radial basis centers $\mathcal{K} = \{K^i \mid i = i..n\}$, the feature vector D has n components where:

$$D_j = \exp(-\beta^i ||X - K^i||^2)$$

where β^i is the inverse variance of rbf center K^i. The radial basis function centers $\mathcal{K}$ are learned in advance with k-means unsupervised learning, using a wide spectrum of logfiles from different environments (see Fig. 4.11). The width of RBF K^i, β^i, is the inverse variance of the points that clustered to it during k-means learning.

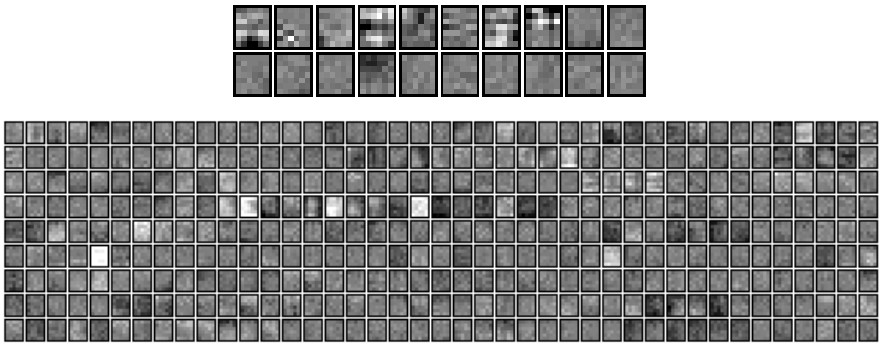


Figure 4.12: The first layer convolutional kernels (*top*) and the second layer convolutional kernels (*bottom*). The filters were trained through supervised training on 500,000 labeled samples taken from 130 logfiles. The labels are obtained by the same stereo-based process described in Section 4.4.

4.5.3 Convolutional Neural Network

Convolutional network architectures and training methods are described in detail in Section 1.3.1. This particular network has two convolutional layers and one subsampling layer. The first convolutional layer has 20 7x6 filters and the second layer has 369 6x5 filters, shown in Figure 4.12. The layers are not fully connected; in particular, Y, U, and V filters are kept separate throughout the 2 layers. The connections between the input and the first layer, and from the first layer to the second, are shown in Figure 4.13. For the purposes of training the network, a final 100 component, fully-connected layer is trained with five outputs. After the network is trained, the fully connected layer is removed such that the *realtime* output of the network is a 100 component feature vector. The filters show that the network is responsive to horizontal structures, such as obstacle feet and other visual boundaries, as well as to general color shifts in the U and V channels (see Fig. 4.12). The network was initialized with random values and trained for 30 epochs using stochastic gradient descent and $L2$ regularization.

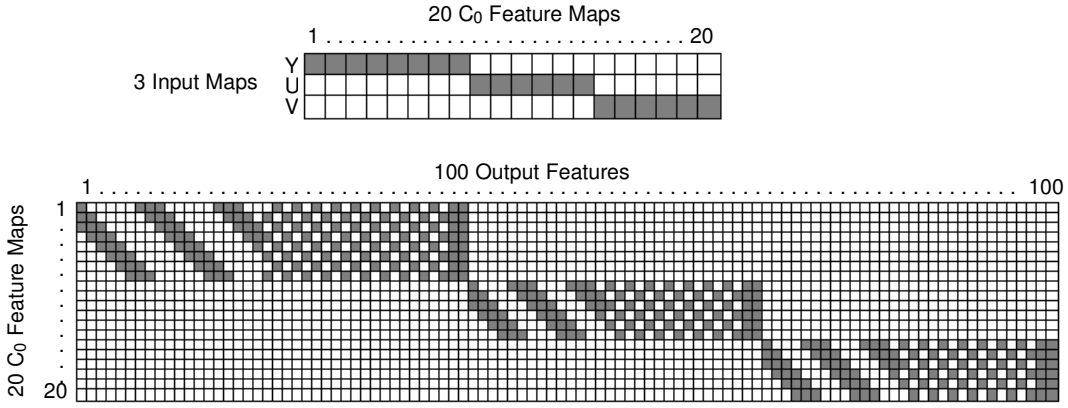


Figure 4.13: The top table shows the connections between the YUV input layers and the 20 feature maps in the first layer. Note that this is not a fully connected network; 8 filters are connected to the Y channel and 6 features are connected to each of the U and V channels. The second layer connections are also shown (*bottom*). The table shows connections between the 20 first layer feature maps and the 100 output features. As in the first layer, connections between Y, U, and V feature maps were separated.

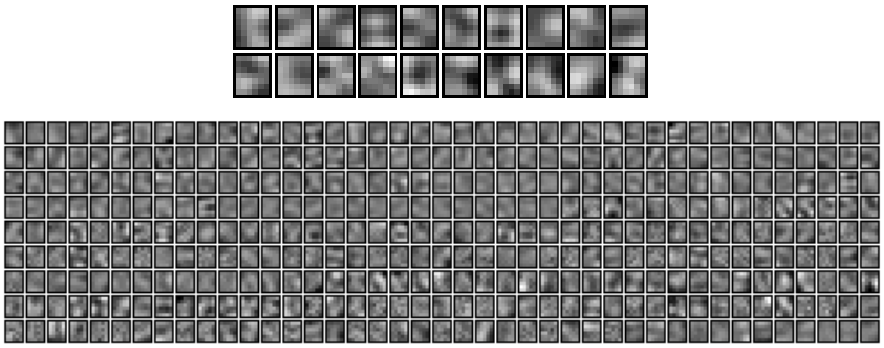


Figure 4.14: Trained filters from both layers of the trained feature extractor. *Top:* the first convolutional layer has 20 7x6 filters. *Bottom:* the second convolutional layer has 369 6x5 filters.

4.5.4 Convolutional Auto-Encoder Network

The third feature extraction approach uses the principles of *deep belief network training* (Hinton et al., 2006; Ranzato et al., 2007c). The basic idea behind deep belief net training is to pre-train each layer independently and sequentially in unsupervised mode using a reconstruction criterion to drive the training. The training architecture and training procedure is given a more detailed exposition in Section 1.3.2.

For this application, the network was built in 2 stages. A set of 20 7x6 filters were first trained using a reconstruction criterion. Then the training samples were transformed with the filters and pooled with a 1x4 max-pooling unit. The resulting feature maps were collected and used to train a second layer of filters (300 6x5 filters), also with a reconstruction criterion. The process produces two sets of filters, which can be used convolutionally to transform an input window into a 100 dimensional feature vector. A two-layer convolutional network is built with the same connection table and kernel dimensions as the auto-encoder filters, and the learned filters are copied into the convolutional network.

The feature maps which are the internal states of the convolutional network are shown in Figure 4.15. The input to the network is a row from the normalization pyramid. The output is a set of 100-dimension feature vectors.

4.5.5 Convolutional Auto-Encoder Tuned with Supervised Training

This is a hybrid approach that marries the advantages of unsupervised auto-encoder training with supervised "fine-tuning". The filters learned through deep belief net training, as described in Section 4.5.4, are used to initialize a convolutional network, which is then re-trained with labeled data and traditional backpropagation. Some of the filters remain very similar, while others are substantially rewritten (see Fig. 4.16). This training scheme can be very beneficial if there is not enough labeled training data to adequately learn the best feature representation. This feature representation outperformed both the purely supervised network and the purely unsupervised auto-encoder network in tests using a groundtruth dataset (see Fig. 6.2).

4.5.6 DrLIM Convolutional Net

This feature extractor was trained using pairwise supervised data and a similarity criterion. The architecture was a two-layer convolutional net with the same connections and kernel dimensions as the supervised convolutional net and the convolutional auto-encoder network. The filters learned by this approach are shown in Fig. 4.17. Using DrLIM training should produce a network with invariance to the transformations present in the matched data samples.

4.5.7 DrLIM Convolutional Net Tuned with Supervised Training

The final feature extractor was trained with a hybrid approach similar to the "fine-tuned" convolutional auto-encoder. The trained DrLIM filters were briefly trained with data from the labeled

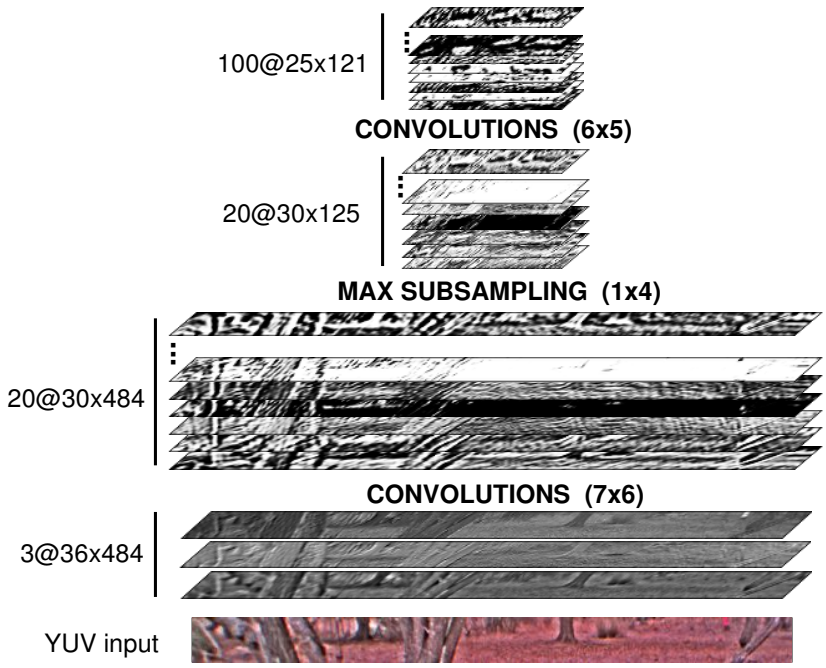


Figure 4.15: The feature maps (the internal states of the network) are shown for a sample input. The input to the network is a variable width, variable height layer from the normalized pyramid. The output from the first convolutional layer is a set of 20 feature maps, the output from the max-pooling layer is a set of 20 feature maps with width scaled by a factor of 4 through pooling, and the output from the second convolutional layer is a set of 100 feature maps. A 3x12x25 window in the input corresponds to a single 100-dimension feature vector in the output.

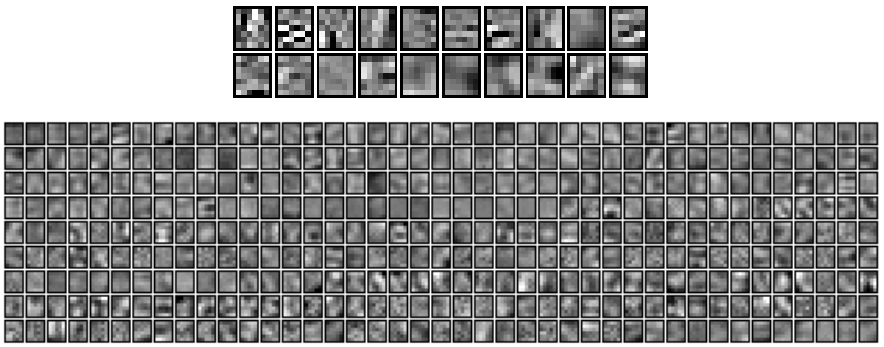


Figure 4.16: The filters were initialized with unsupervised auto-encoder training, followed by limited top-down supervised training.

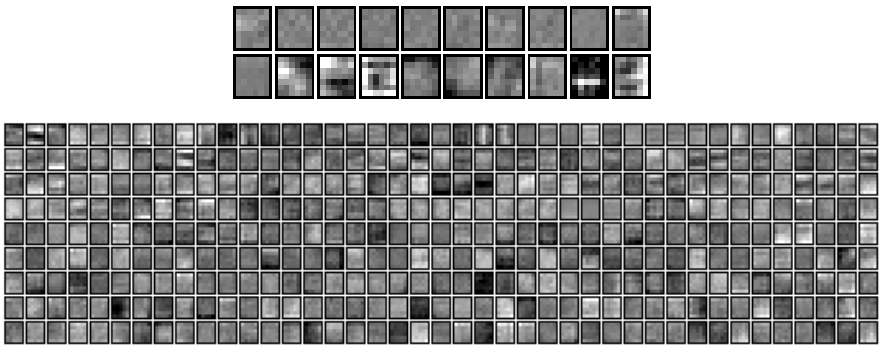


Figure 4.17: These filters were trained with a co-location pairwise similarity criterion and the DrLIM loss function. They contain many strong horizontal and vertical edge detectors, but the Y-channel filters (the top row of the top layer) are very flat.

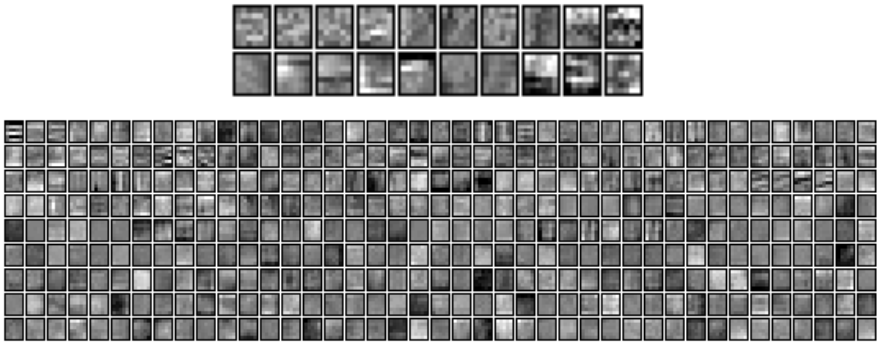


Figure 4.18: These filters reflect hybrid training: the network was initialized with the DrLIM filters from the previous section, then trained with supervision for a short time (1 epoch). The first layer filters are significantly re-wired by this approach.

dataset used to train the supervised convolutional net. The filters are shown in Fig. 4.18). This feature extractor produced the best results on the groundtruth dataset.

4.6 Realtime Terrain Classification

The online learning framework takes the feature vectors and supervisory labels and trains 5 binary classifiers. Since the number of labeled training samples in each category can vary widely (open lawn vs. dense forest, for example), we use 5 ring buffers to accumulate up to 1000 training samples of each category. This not only balances the training between the multiple classes, but also acts as a rudimentary short-term memory: we train on several frames worth of data at each timestep, so we "remember" obstacles and groundtypes for at least 2 timesteps, and as many as 30 timesteps, after our last direct sighting of them. Data is removed from its ring buffer if it persists for more than 20 timesteps; this prevents overtraining.

Since the online classifier trains from and then classifies every frame that it receives, it must be simple and efficient. A separate logistic regression is trained on each of the 5 categories,

using a modified one-against-the-rest training strategy in which overlapping categories are not trained discriminatively against each other - i.e., super-traversable and traversable samples are not presented to the same classifier as positive and negative examples, nor are super-obstacle and obstacle. For a feature vector $\mathbf{x}$, we compute the output of each regression i through its weight vector, bias, and a logistic sigmoid function:

$$q_i = f(\mathbf{w_i}\mathbf{x} + b), \quad \text{where} \quad f(z) = \frac{1}{1 + e^{-z}}$$

The labeled feature vectors can be used to train the regressions by minimizing a loss function. The loss function that is minimized is the Kullback-Leibler divergence or relative entropy:

$$D_{KL}(P||Q) = \sum_{i=1}^{K} p_i \log p_i - \sum_{i=1}^{K} p_i \log q_i,$$

where p_i is the probability that the sample belongs to class i, as given by the stereo supervisor labels. q_i is the classifier's output for the probability that the sample belongs to class i. The loss for each binomial regression is

$$Loss_i = -p_i \log q_i - (1 - p_i) \log(1 - q_i)$$

The weights of each regression are updated using stochastic gradient descent, since gradient descent provides strong regularization over successive frames and training iterations. The gradient update for the ith regression with training sample $\mathbf{x}$ is

$$\Delta \mathbf{w_i} = -\eta \frac{\partial Loss}{\partial \mathbf{w_i}} = -\eta \left(p_i - q_i\right) \mathbf{x}$$

The regressions are trained, using all the samples in the ring buffer, for one epoch on each timestep. After training, all inputs from the current frame are labeled by all 5 regressions, yielding a 5 component likelihood vector for each input. Even the stereo-labeled inputs are labeled

with the trained regressions, since the resulting classifications are often smoother and more accurate than the training labels. The output of the long-range module, after training and classification, consists of a set of points given in vehicle-relative coordinates and the associated likelihood vectors.

5

MAPPING FROM LONG-RANGE VISION WITH LABEL AND RANGE UNCERTAINTY

Joint work with Pierre Sermanet.

5.1 Introduction

A long-range traversability classifier such as the one described in the previous chapter will often exhibit uncertainty, evidenced by changing traversability predictions for a single location or by low likelihood values in the classifier output. If the classifier outputs for a particular object or area of terrain are collected over multiple timesteps, normal fluctuations in the label classification will be observed. Label uncertainty has several causes:

- As the robot navigates, the visual appearance of a single object may change quickly, for instance because of lighting change as the robot passes through a deep shadow. Although the learning will adapt to the change, there may be a short period of poor labels.

- The learning is guided wholly by the stereo supervisor. The classifier is quite robust to noisy labels from the stereo module, but if the stereo labels are very bad or simply lacking (from glare or deep shadow or repeating textures), the learning will eventually falter as well.

- The classifier may give an incorrect label to a far distant area that is substantially dissimilar to nearby terrain, then give a correct label once it is closer or has seen a similar example at close range.

Label uncertainty, while normal from a machine learning point of view, is deleterious for the robot because it leads to oscillations in planning and poor driving. We propose an aggregation scheme to accommodate and model label uncertainty in a cost map.

Range uncertainty must also be accommodated in a navigation system with long-range vision. Errors in range estimation occur when the classifier target is beyond stereo range and the ground plane is not consistent from near to far range. Objects that are above the ground plane - trees, buildings, etc. - will also have poor distance estimates because the object will be incorrectly projected far in the distance if there are no stereo cues. We propose a mapping geometry that accurately reflects the range uncertainty associated with image-plane obstacle labeling. Furthermore, the map can represent an effectively infinite radius with a finite number of cells.

This chapter discusses mapping in the context of the uncertainties of long-range vision; i.e., the aggregation of labels over time and the quasi-image-space geometry of the map. Further details of the hyperbolic-polar mapping can be found in (Sermanet et al., 2008).

5.2 Mapping: Histograms

Probabilistic approaches (Elfes, 1991) and fuzzy logic (Oriolo et al., 1997) methods have been applied to the problem of mapping with label uncertainty, as well as trivial approaches such as eliminating memory (overwriting old labels) or maintaining running averages. We use a histogram approach suited for multiple classes which allows the traversability decision to be delayed until planning time. Each cell in the map contains k bins, each bin corresponding to a terrain class (see Fig. 5.1). The long-range classifier computes a vector of likelihoods for each window in the image. The vector is associated with a particular robot-relative location, which is associated with a cell in the map. The vector is merged with the existing histogram of this cell through a simple addition. Before planning, the histogram is translated into a traversability cost

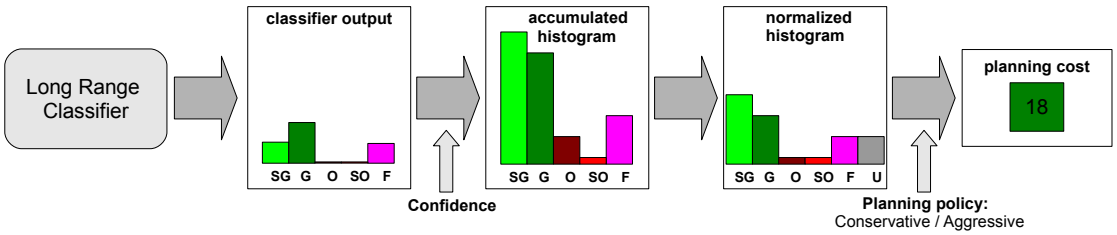


Figure 5.1: **Histogram to cost process.**

The output of the classifier is multiplied by the current learning confidence and added to an existing histogram. Before converting the histogram to a planning cost, it is normalized. Finally, the current planning policy converts the normalized histogram to a single planning cost.

(see Fig. 5.2). At that time, the traversability decision can be modulated by the current planning policy: conservative vs. aggressive. As label uncertainty increases with distance, a distance decay must be applied to incoming frames.

5.2.1 Histogram to Cost Transformation

Let $\sum_k bin_k + c$ be the sum of all bins of a histogram with an added constant c which forces the confidence towards 0 when the sum is very small, thus limiting the confidence when the target area has only been seen briefly. This sum is used for normalization. A set of weights w is tuned, one for each terrain class. The normalized weighted sum S scaled by a gain parameter γ is defined by:

$$S = \gamma \frac{\sum_k w_k bin_k}{\sum_k bin_k + c}$$

S is then used to compute the planning cost $cost$ using the piecewise linear function shown in Fig. 5.2:

if $(S <= 0)$ **then**

$cost = cost_{unexplored} + S * (cost_{unexplored} - cost_{min})$

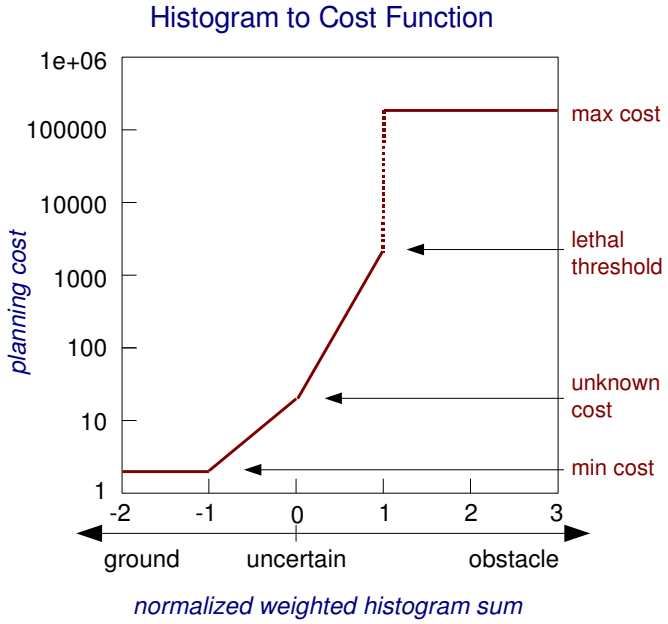


Figure 5.2: **Planning cost mapping from normalized histogram sum.**

A histogram sum of 0 represents complete uncertainty, sums of -1 and below are given the minimum cost, and sums of 1 or higher are given the maximum cost.

else

if $(S <= 1)$ **then**

$cost = cost_{unexplored} + S * (cost_{lethal} - cost_{unexplored})$

else

$cost = cost_{lethal}$

end if

end if

$cost = \max(cost, cost_{min})$

5.3 Mapping: Geometry

In the image plane, a single pixel covers a wildly varying range of distances. A single pixel on nearby ground (near the bottom of the image) covers a few centimeters, while a pixel near the horizon covers an very large range. For a flat ground plane, the mapping of image-plane pixels to distances is hyperbolic from the bottom of the image to the horizon. For this reason, we propose to represent the environment through a robot-centered *hyperbolic-polar map* (h-polar). This representation allows to map the entire world to a finite number of cells, while being faithful to the type of uncertainty afforded by image-plane labels. Figure 5.3 shows the geometry of the hpolar map.

Previous work on robot-centered, non-uniform mapping include log-polar representations (Longega et al., 2003) and multi-resolution grid-based maps (Behnke, 2004). The main advantage of the h-polar approach over these methods is the ability to represent an effectively infinite radius with a finite number of cells. Even more important is a representation of range uncertainty that directly corresponds to that associated with image-plane labeling. Pure image-plane labeling and planning (Zhang and Ostrowski, 2002), or visual motion planning, presents the advantage of being free of expensive transformations and pose errors, but can only operate with one single frame at the time.

Planning is certainly a critical part of any navigation system, and this one is no exception. The planning in the hpolar map is done using Dijkstra's shortest path algorithm, with a tree-based mediation between candidate routes in the long-range map and candidate routes in the short-range map. The planning processes go well beyond the scope of this work; for details, see (Sermanet et al., 2008).

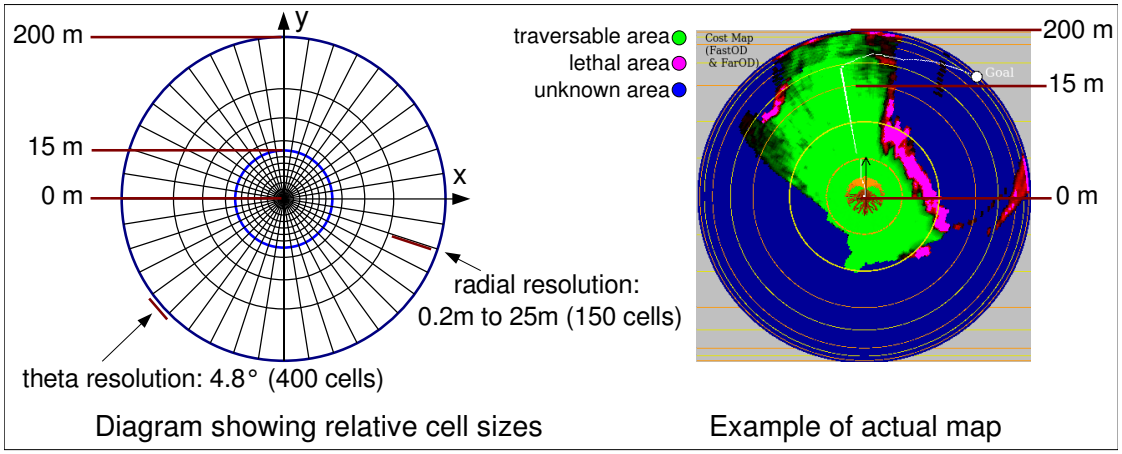


Figure 5.3: The relative sizes of cells in the hyperbolic map are shown on the left. In the center of the map, from 0 to 15 meters, there are 75 rows of cells with a fixed 20 cm radial resolution. From 15 meters to 200 meters there are 75 hyperbolic rows with resolution that grows from 20 cm to 25 meters. The last row has an infinite radial resolution.

6
RESULTS AND DISCUSSION

6.1 Results

We have tested the long-range vision classifier independently as well as testing its effect on the full navigation system. Independent testing of the classifier is difficult, because the stereo supervision labels that would normally be used to judge whether the classifier is well-trained are extremely short-range and often noisy. In order to give a truer estimate of the accuracy of the classifier, we created a groundtruth data set containing 160 hand-labeled frames scattered over 25 logfiles.

6.1.1 Ground Truth and Stereo Error

To build a groundtruth data set for offline testing, a human operator hand-labels several frames from each file in a collection of logs, using a GUI that was build to facilitate the process. The labeling is binary: the human draws lines to demarcate obstacle foot lines or the boundaries of other lethal areas, and every pixel from the bottom of the image to the first such "lethal line" is assumed to be traversable (see Fig. 6.1. To test a particular classifier configuration using the groundtruth data, the long-range vision module is run in simulation mode on the groundtruth set of log files, and when a labeled frame comes up the error is computed over the frame. The error depends on the difference in pixel labels between identical areas in the groundtruth frame and the classified frame. Since the groundtruth has binary labels but the classifier has 5 labels, the classifier output is binarized (obstacle, super-obstacle, and footline indicate lethality, while ground and super-ground indicate traversable terrain) before calculating the error.

Fig. 6.2 shows the groundtruth error of the long-range classifier. The 25 groundtruth logfiles

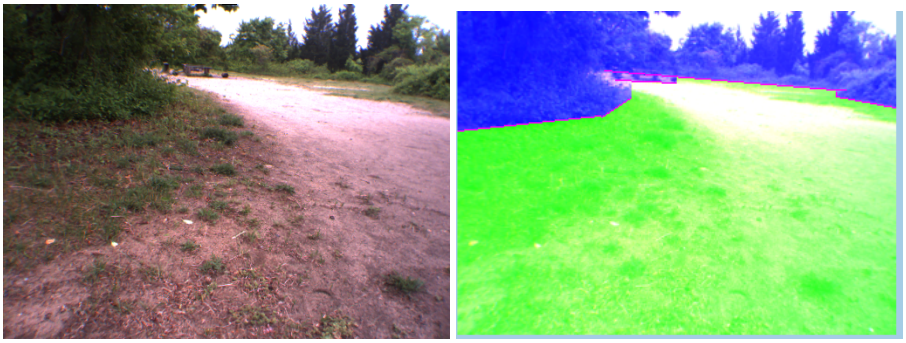


Figure 6.1: Groundtruth images have been labeled using a binary segmentation by a human, as in this example

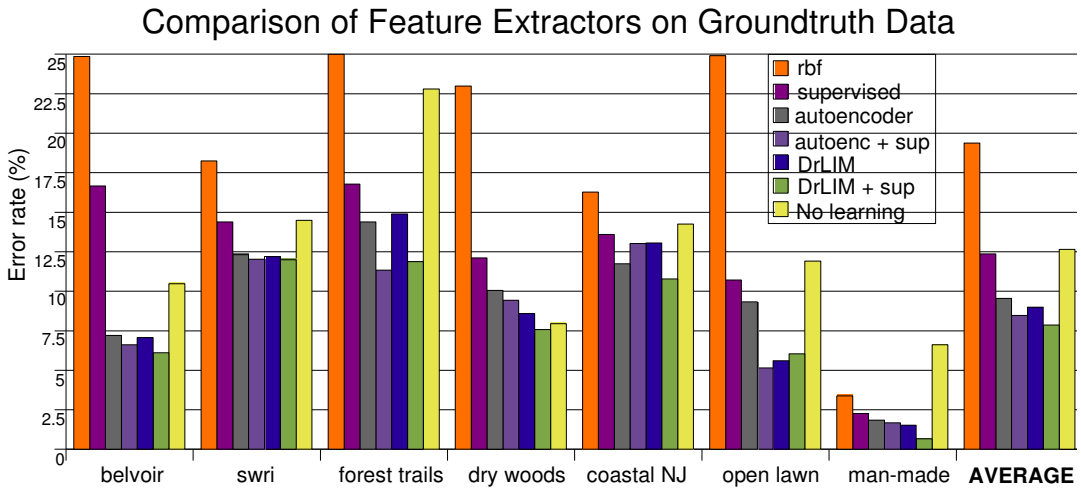


Figure 6.2: Error rates are given for a groundtruth data set. The logfiles in the data set are grouped into 7 sets by their dominant terrain and location. In each chart, 7 different feature extractors are compared: RBF features, supervised convolutional net, convolutional auto-encoder (unsupervised), hybrid auto-encoder (initialized with auto-encoder, tuned with supervision), DrLIM convolutional, DrLIM hybrid (initialized with DrLIM, tuned with supervision), and the DrLIM hybrid with no online learning (default weights only).

are divided into 7 groups according to the terrain and geography of the logfiles. At 12.36% error, the supervised network does not perform well on the groundtruth data, probably because of the wider data distribution found in the groundtruth dataset. The 2 purely unsupervised representations, auto-encoder and RBF, perform quite differently. The RBF is slower to adapt in general and has poor accuracy on visually complex terrain (19.37% average error). The auto-encoder does very well, but is outperformed by the hybrid version with supervised fine-tuning (9.55% vs. 8.46%). Apparently the supervision added just the patterns that the unsupervised features were lacking, allowing a stronger classification rate while retaining the generality of the unsupervised features. The same relationship holds for the 2 weakly supervised networks. The pure DrLIM does well at 8.99%, but the hybrid DrLIM, fine-tuned with supervision, has the best overall error rate of 7.86%. The last bar in the figure shows the performance of the hybrid autoencoder, but tested without training the classifier online. Instead the classifier weights were kept fixed on a set of general default weights. Not surprisingly, the no-learning classifier has difficulty on all the logfiles and at 12.64% has the second worst average error.

To quantify the accuracy of the stereo module, it was tested against the groundtruth set and found to be quite erroneous. In fact, it was less accurate, overall, than the classifier performance - surprising, since the classifier relies on the stereo module as its only source of training data. The difference in error rates for stereo module vs. classifier are shown in Figure 6.3. The positive data points represent frames where the classifier had a lower error rate than the training data. The classifier's improvement over the training data implies that there is noise in the training labels that is being smoothed, or regularized, by the classifier. We can also conclude that the groundtruth labels accord well with visual cues in the image. This is not surprising, of course, since the human groundtruth labeler has nothing *but* visual cues on which to base her labels.

Fig. 6.4 shows seven examples of long-range classifications in very different terrain. The input image, the stereo labels, and the classifier outputs are shown in each case. Note that

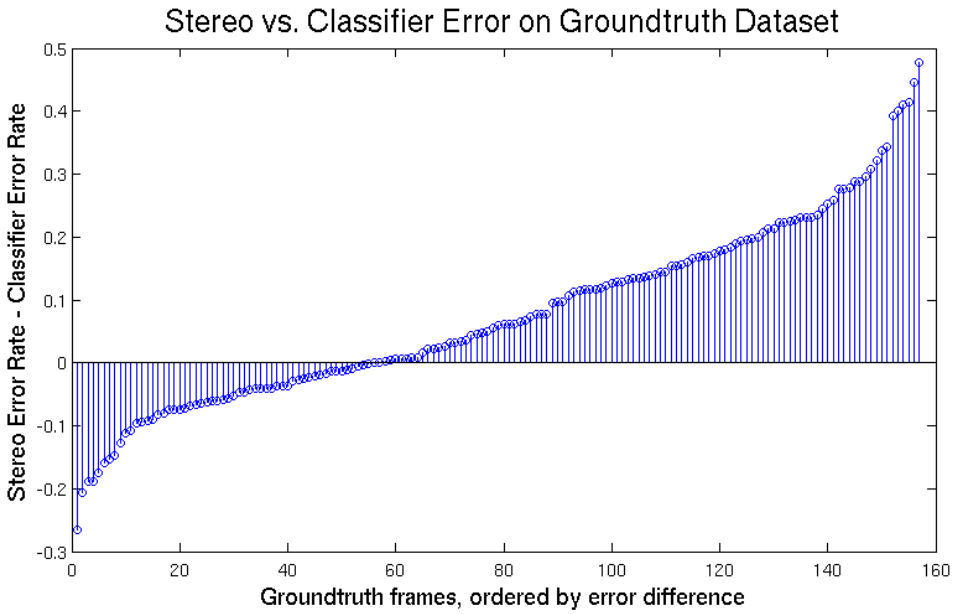


Figure 6.3: The difference between stereo error and classifier error is plotted, showing that the online classifier has higher accuracy than its own training data on a majority of the groundtruth frames. The positive data points represent frames where the classifier had a lower error rate than the training data.

generally the classifier output gives a better labeling even for the part of the image that can be labeled by stereo. The far-range image portion is smoothly labeled as well. In contrast to a color-based classifier, this classifier is able to recognize many different complex objects or groundtypes in the same scene.

Figures 6.5, 6.6, 6.7, 6.8, 6.9, and 6.10 show examples of long-range mapping with a hyperbolic polar map, as described in Chapter 5. Each example shows the left and right input frames, the output map with short-range (10 meter) stereo vision, and the output map with long-range classifier outputs. The planned long-range route to the goal can be seen in each of the examples; the route and the goal are both white.

6.1.2 Full System Field Experiments

The long-range vision system has been used extensively in the full navigation system built on the LAGR platform. It runs at 2 Hz, which is too slow to maintain good close-range obstacle avoidance, so the system architecture runs 2 processes simultaneously: a fast, low-resolution stereo-based obstacle avoidance module and planner run at 8-10 Hz and allow the robot to nimbly avoid obstacles within a 5 meter radius. Another process runs the long-range vision and long-range planner at 2 Hz, producing strategic navigation and planning from 5 meters to the goal.

We present experimental results obtained by running the robot on 4 courses with the long-range vision turned on and turned off. With the long-range vision turned off, the robot relies on its fast planning process and can only detect obstacles within 5 meters. Course 1 (see Fig. 6.11 and Table 6.15) is a narrow wooded path that proved very difficult for the robot with long-range vision off, since the dry scrub bordering the path was difficult to see with stereo alone. The robot had to be rescued repeatedly from entanglements off the path. With long-range vision on, the robot saw the scrub and path clearly and drove cleanly down the path to the goal. Course 2 (see Fig. 6.12 and Table 6.15) was a long wide path with a clearing to the north that had no outlet - a large natural cul-de-sac. Driving with long-range vision on, the robot saw the long path and drove straight down it to the goal without being tempted by the cul-de-sac. Driving without long-range vision, the robot immediately turned into the cul-de-sac and became stuck in scrub, needing to be manually driven out of the cul-de-sac and restarted in order to reach the goal. Course 3 (see Fig. 6.13 and Table 6.15) was a short but complex path from one clearing in the scrub to another such clearing. Although the paths taken by the short-range and long-range systems are similar, the path of the long-range system is smoother and avoids detouring down the false path. Course 4 compared the full long-range system against the CMU baseline system, which uses no learning in its 10 meter stereo-vision navigation. To reach the goal, the robot

had to navigate around a large tall-grass barrier (see Fig 6.14). The long-range system saw the impassable grass from far away and planned around the barrier. The baseline system navigated straight to the barrier, then turned and felt along the edge, at times getting snagged in the taller grass or turning in circles.

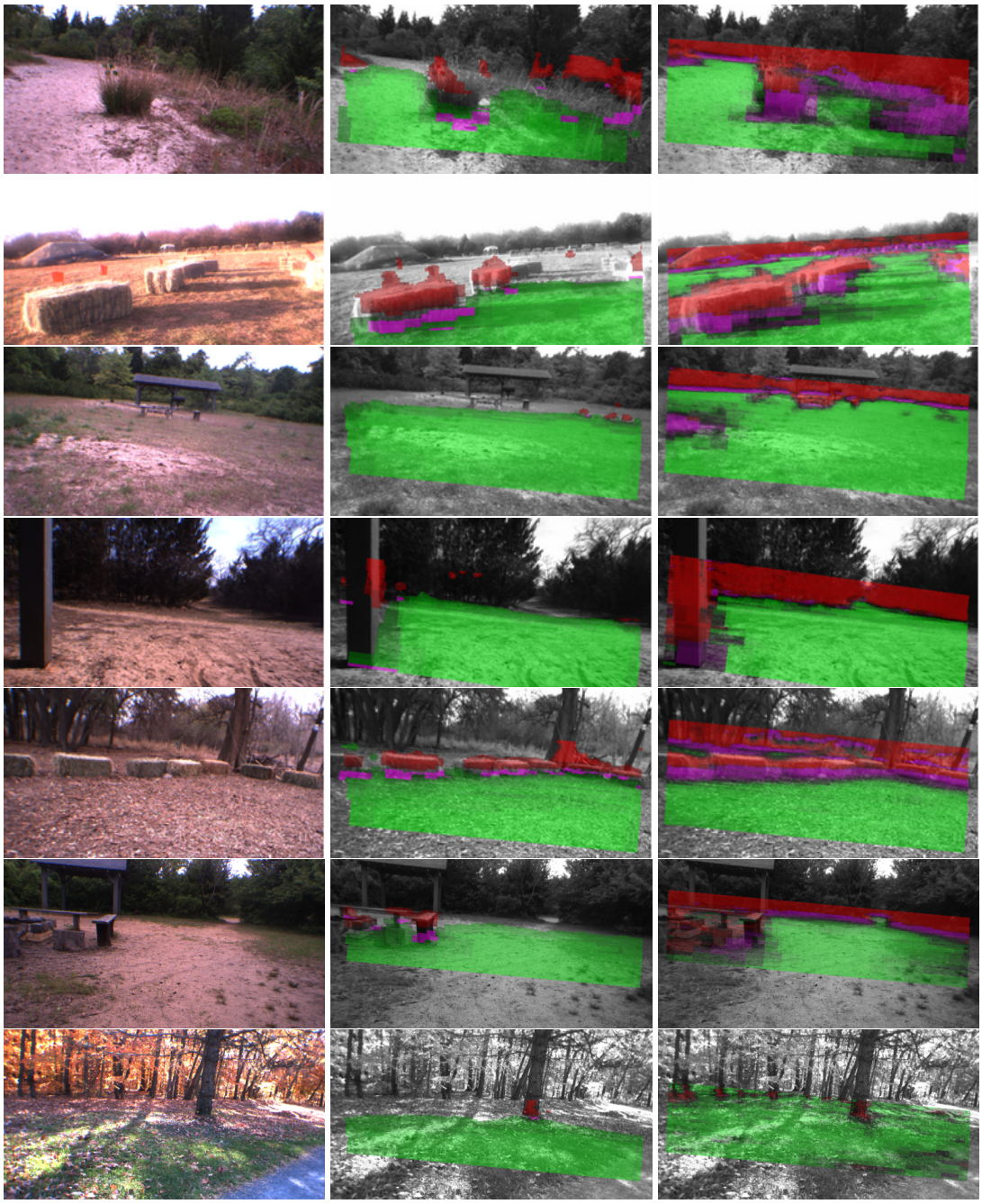


Figure 6.4: Qualitative examples of the success of the long-range classifier in different terrain. **Left**: RGB input; **middle**: training labels; **right**: classifier output. Green is traversable, red is obstacle, and pink is footline.

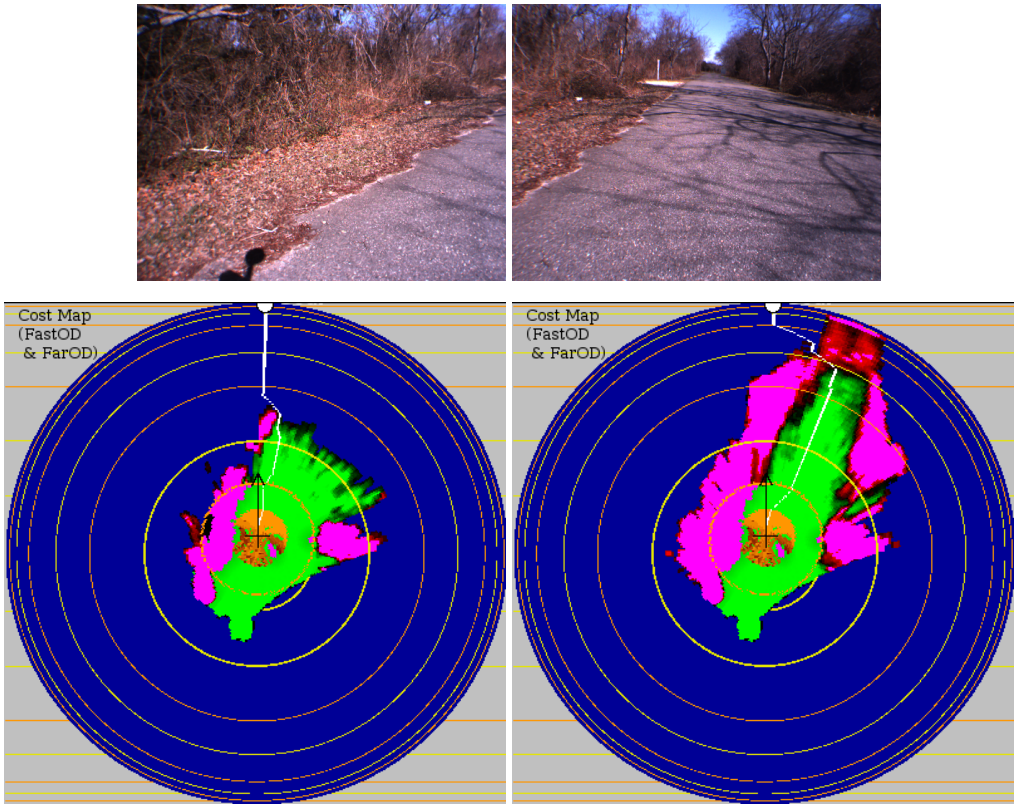


Figure 6.5: *Left: short-range; Right: long-range.*

Course 1: On this course, the short-range system leaves the road and enters a long cul-de-sac to the left, while the long-range system proceeds down the road to the goal.

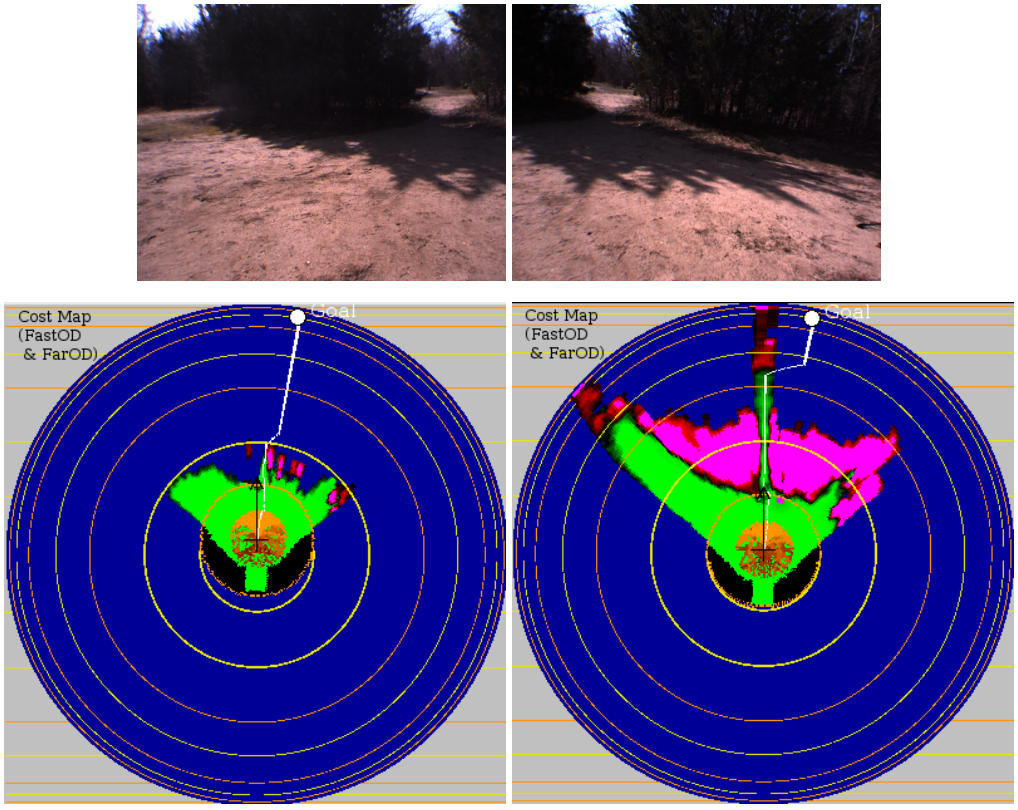


Figure 6.6: *Left: short-range; Right: long-range.*

Course 2: Although both the short- and long-range systems find the goal, the long-range system is more robust to error and stays on track better than the short-range.

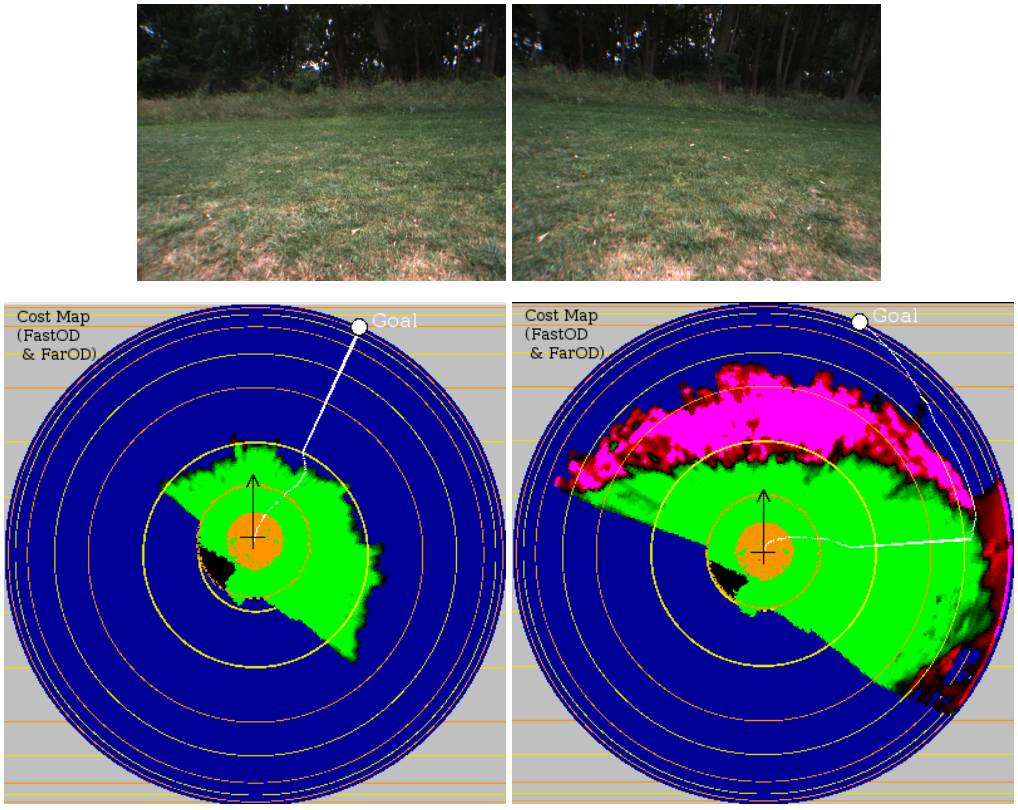


Figure 6.7: *Left: short-range; Right: long-range.*

Course 3: This shot is looking a very long, dense wall of trees. There is a long path around the forest to the goal, but the short-range does not see it.

Figure 6.8: *Left: short-range; Right: long-range.*

Course 4: This is the beginning of a long road leading around some trucks and buildings. The long-range classifier sees the route around the building from the beginning.

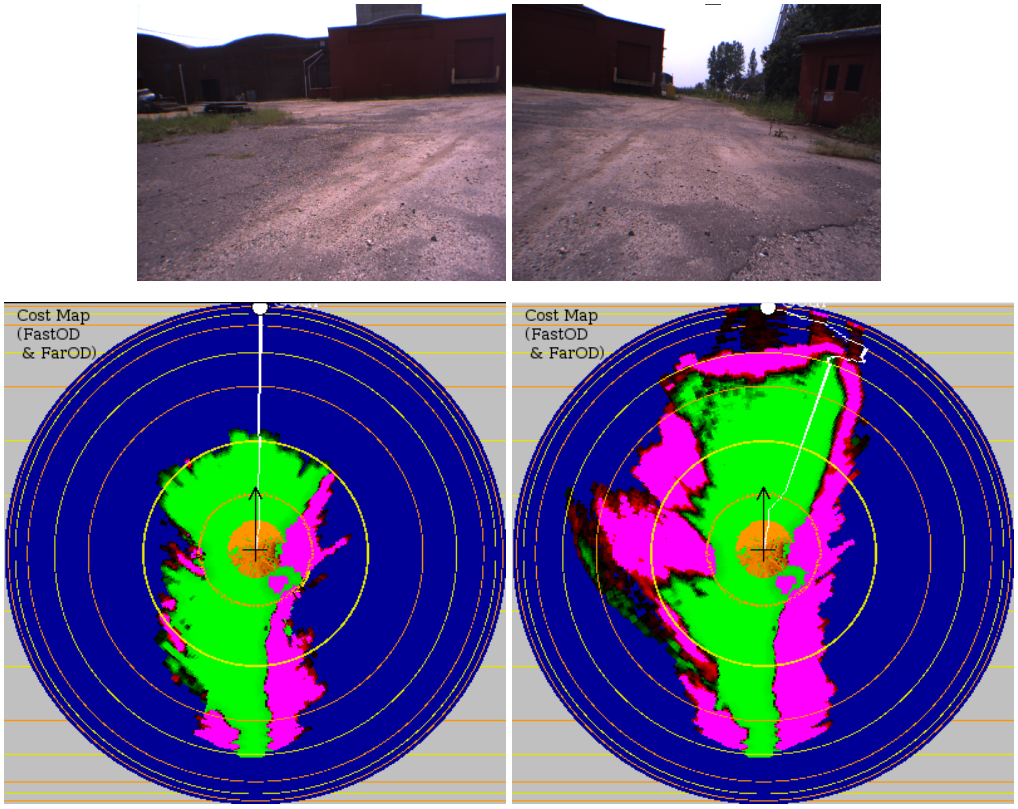


Figure 6.9: *Left: short-range; Right: long-range.*

Course 5: This is the same course as the previous one, but closer to the buildings. The long-range system still wants to navigate around the building, and the short-range still does not see any obstacle.

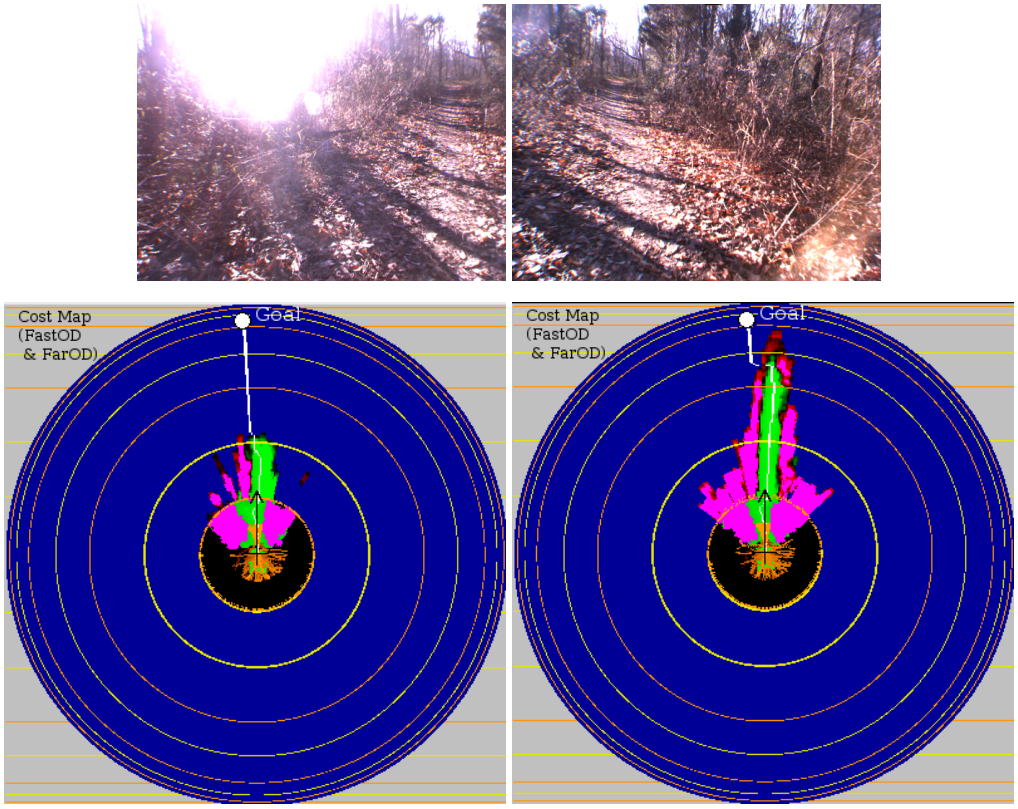


Figure 6.10: *Left: short-range; Right: long-range*

Course 6: This is a wooded path, overgrown and full of errors for the stereo supervisor (sun flares, sparse branches). Without the long-range, the system tends to take ill-advised short-cuts off the path and get stuck.

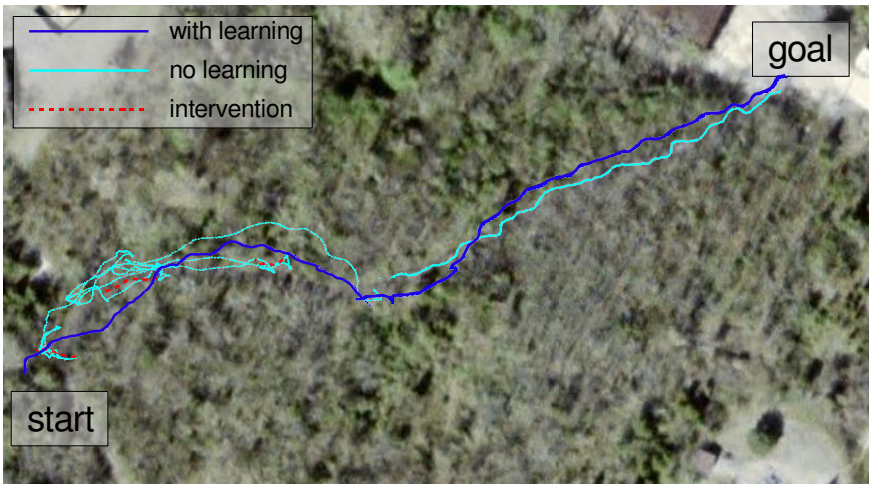


Figure 6.11: This was a long course through a scrubby, dense wood, following a narrow rutted path. Without long-range learning, the robot had a very difficult time staying on the path and had to be rescued from the bushes 3 times.

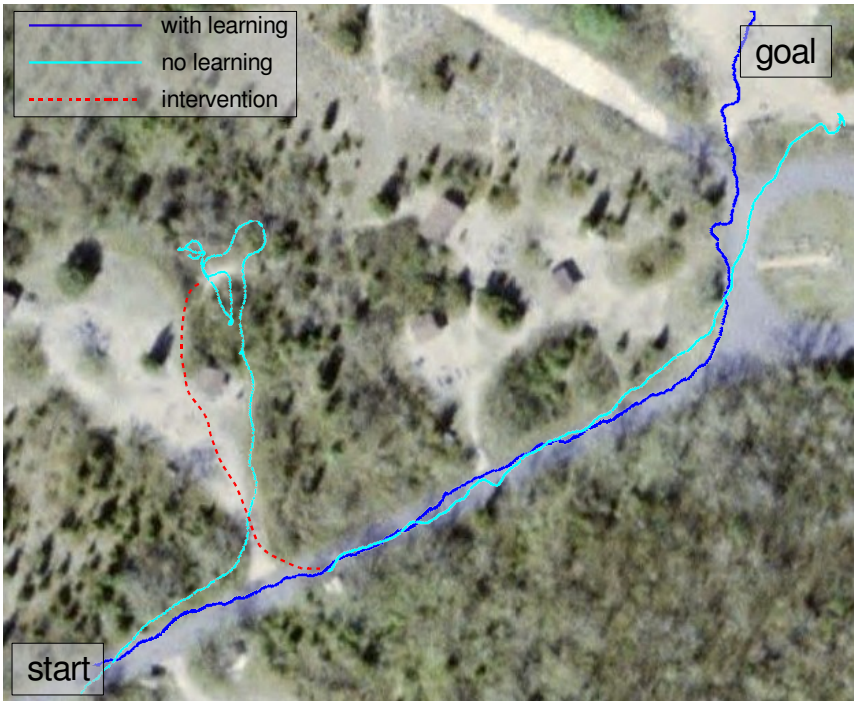


Figure 6.12: This course led down a wide paved road, with a natural cul-de-sac. The non-learning system took the turn into the cul-de-sac and had to be rescued. The long-range system could easily see the road and did not enter the cul-de-sac.

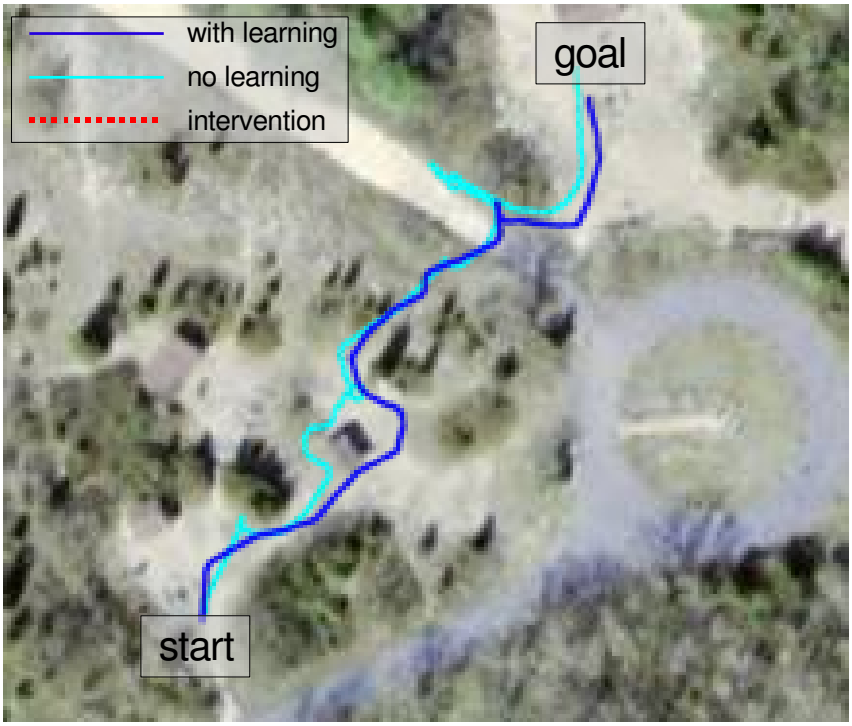


Figure 6.13: The third course zigzagged through 3 picnic areas. The paths taken by the different systems are similar, but the long-range vision path is smoother and optimized.

Figure 6.14: This experiment compared the long-range system with the CMU baseline software. The fourth course was much longer, and had a large tall-grass obstacle (the green terrain in the satellite picture was impassable tall grass when the test occurred). The short-range baseline system drove straight to the obstacle before turning and navigating around it, whereas the long-range system detected the barrier and planned around it, jumping onto a paved path for a short distance.

Course 1	Total Time	Total Distance	Inter-ventions
Short-Range	321 sec	271.9 m	3
Long-Range	155.5 sec	166.8 m	0
Course 2			
Short-Range	196.1 sec	207.5 m	1
Long-Range	142.2 sec	165.1 m	0
Course 3			
Short-Range	123.7 sec	122.9 m	0
Long-Range	108.7 sec	113.8 m	0
Course 4			
Baseline	503.77 sec	479.16 m	1
Long-Range	254.2 sec	313.97 m	0
Average improvement of Long over Short	**173.2%**	**142.3%**	**4/0**

Figure 6.15: Time-to-goal and Distance-to-Goal metrics for 4 offroad courses.

CONCLUSION

We have described, in detail, an self-supervised learning approach to long-range vision in off-road terrain. The classifier is able to see smoothly and accurately to the horizon, identifying trees, paths, man-made obstacles, and ground at distances far beyond the 10 meters afforded by the stereo supervisor. Complex scenes can be classified by our system, well beyond the capabilities of a color-based approach. The success of the classifier is due to the use of large context-rich image windows as training data, and to the use of an invariant, weakly supervised network for learned feature extraction. The accuracy of the classifier has been shown through systemic field testing as well as through comparison with a hand-labeled groundtruth dataset.

Although the perception system does not satisfy all of the requirements set forth in the introduction for intelligent, adaptive, mobile perception, we believe it does provide some of the necessary components for robot perception, the most important being a high-level scale-invariant feature representation as a basis for learning and a near-to-far learning strategy. The success of the visual classifier allows the robot to drive strategically, planning ahead for obstcles and shooting towards distant paths. When it wakes up in novel terrain, it adapts within seconds to the new patterns, colors, and shapes.

BIBLIOGRAPHY

Adelson, E., Anderson, C., Bergen, J., Burt, P., and Ogden, J. (1984). Pyramid methods in image processing. *RCA Engineer*, 29(6).

Angelova, A., Matthies, L., Helmick, D., and Perona, P. (2007). Dimensionality reduction using automatic supervision for vision-based terrain learning. In *Proc. of Robotics: Science and Systems (RSS)*.

AT&T Faces Database. http://www.uk.research.att.com/facedatabase.html.

Behnke, S. (2004). Local multiresolution path planning. In *Robocup 2003: Robot Soccer World Cup VII*.

Belhumeur, P., Hespanha, J., and Kriegman, D. (1997). Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 19(7).

Belkin, M. and Niyogi, P. (2003). Laplacian eigenmaps and spectral techniques for embedding and clustering. *Advances in Neural Information Processing Systems (NIPS)*, 15(6):1373–1396.

Bellutta, P., Manduchi, R., Matthies, L., Owens, K., and A.Rankin (2000). Terrain perception for Demo III. *Intelligent Vehicle Symposium*.

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2006). Greedy layer-wise training of deep networks. In *Advances in Neural Information Processing Systems (NIPS)*. MIT Press.

Bengio, Y. and LeCun, Y. (2007). Scaling learning algorithms towards AI. In Bottou, L.,

Chapelle, O., DeCoste, D., and Weston, J., editors, *Large-Scale Kernel Machines*. MIT Press.

Bengio, Y., Paiement, J., and Vincent, P. (2004). Out-of-sample extensions for LLE, Isomap, MDS, Eigenmaps, and spectral clustering. In *Advances in Neural Information Processing Systems (NIPS)*, volume 16. MIT Press.

Broggi, A., Caraffi, C., Cattani, S., and Fedriga, R. (2006). A decision network based framework for visual off-road path detection problem. In *In Proc. of International Conference on Intelligent Transportation Systems*, pages 951–956.

Bromley, J., Guyon, I., LeCun, Y., Sackinger, E., and Shah, R. (1993). Signature verification using a Siamese time delay neural network. In *Advances in Neural Information Processing Systems (NIPS)*. MIT Press.

Buck, L. and Axel, R. (1991). A novel multigene family may encode odorant receptors: a molecular basis for odor recognition. *Cell*, 65:175–187.

Chopra, S., Hadsell, R., and LeCun, Y. (2005). Learning a similarity metric discriminatively, with application to face verification. In *Proc. of Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE.

Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: deep neural networks with multitask learning. In *Proc. of International Conference on Machine Learning (ICML)*. ACM.

Cox, T. and Cox, M. (1994). Multidimensional scaling. *London: Chapman and Hill*.

Dahlkamp, H., Kaehler, A., Stavens, D., Thrun, S., and Bradski, G. (2006). Self-supervised

monocular road detection in desert terrain. In *Proc. of Robotics: Science and Systems (RSS)*. MIT Press.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Proc. of Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 886–893. IEEE.

DeSouza, G. and Kak, A. (2002). Vision for mobile robot navigation: A survey. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 24(2):237–267.

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

Dima, C., Vandapel, N., and Hebert, M. (2004b). Classifier fusion for outdoor obstacle detection. In *Proc. of International Conference on Robotics and Automation (ICRA)*. IEEE.

Donoho, D. and Grimes, C. (2003). Hessian eigenmap: Locally linear embedding techniques for high dimensional data. *Proceedings of the National Academy of Arts and Sciences*, 100:5591–5596.

Duda, R. and Hart, P. (1972). Use of the Hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15.

Elfes, A. (1991). Occupancy grids: A stochastic spatial representation for active robot perception. In *Autonomous Mobile Robots: Perception, Mapping, and Navigation (S. S. Iyengar and A. Elfes, eds.), IEEE Computer Society Press, pp. 6071*.

FERET Faces Database. http://www.itl.nist.gov/iad/humanid/feret/.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*.

Goldberg, S., Maimone, M., and Matthies, L. (2002). Stereo vision and robot navigation software for planetary exploration. In *Proc. of IEEE Aerospace Conference*.

Goldberger, J., Roweis, S., Hinton, G., and Salakhutdinov, R. (2005). Neighbourhood components analysis. In *Advances in Neural Information Processing Systems (NIPS)*. MIT Press.

Grudic, G. and Mulligan, J. (2006). Outdoor path labeling using polynomial Mahalanobis distance. In *Proc. of Robotics: Science and Systems (RSS)*. MIT Press.

Grudic, G., Mulligan, J., Otte, M., and Bates, A. (2007). Online learning of multiple perceptual models for navigation in unknown terrain. In *International Conference on Field and Service Robotics*.

Hadsell, R., Chopra, S., and LeCun, Y. (2006a). Dimensionality reduction by learning an invariant mapping. In *Proc. of Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE.

Hadsell, R., Erkan, A., Sermanet, P., Ben, J., Kavukcuoglu, K., Muller, U., and LeCun, Y. (2007a). A multi-range vision strategy for autonomous offroad navigation. In *Proc. of Robotics and Applications (RA)*.

Hadsell, R., Sermanet, P., Ben, J., Erkan, A., Han, J., Flepp, B., Muller, U., and LeCun, Y. (2007b). Online learning for offroad robots: Using spatial label propagation to learn long-range traversability. In *Proc. of Robotics: Science and Systems (RSS)*. MIT Press.

Hadsell, R., Sermanet, P., Ben, J., Han, J., Chopra, S., Ranzato, M., Sulsky, Y., Flepp, B., Muller, U., and LeCun, Y. (2006b). On-line learning of long-range obstacle detection for offroad robots. The Learning Workshop at Snowbird.

Hamner, B., Singh, S., and Scherer, S. (2006). Learning obstacle avoidance parameters from operator behavior. *Journal of Field Robotics*, 23(11-12).

Happold, M., Ollis, M., and Johnson, N. (2006). Enhancing supervised terrain classification with predictive unsupervised learning. In *Proc. of Robotics: Science and Systems (RSS)*. MIT Press.

Hinton, G., Osindero, S., and Teh, Y. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554.

Hinton, G. and Roweis, S. (2004). Stochastic neighbor embedding. In *Advances in Neural Information Processing Systems (NIPS)*. MIT Press.

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

Hoiem, D., Efros, A., and Hebert, M. (2005). Geometric context from a single image. In *Proc. of Int'l. Conference of Computer Vision (ICCV)*. IEEE.

Hoiem, D., Efros, A., and Hebert, M. (2006a). Putting objects in perspective. In *Proc. of Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE.

Hoiem, D., Efros, A., and Hebert, M. (2006b). Recovering surface layout from an image. *International Journal of Computer Vision (IJCV)*.

Hong, T., Chang, T., Rasmussen, C., and Shneier, M. (2002). Road detection and tracking for autonomous mobile robots. In *Proc. of SPIE Aeroscience Conference*.

Hsuan Yang, M., Ahuja, N., and Kriegman, D. (2000). Face recognition using kernel eigenfaces. *In Proc. of the 2000 IEEE International Conference on Image Processing (ICIP)*, 1:37–40.

Hua, G., Brown, M., and Winder, S. (2007). Discriminant embedding for local image descriptors. In *Proc. of Int'l. Conference of Computer Vision (ICCV)*. IEEE.

Hubel, D. and Wiesel, T. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *Physiology*.

Huertas, A., Matthies, L., and Rankin, A. (2005). Stereo-based tree traversability analysis for autonomous off-road navigation. In *Workshop of Applications of Computer Vision*. IEEE.

Iagnemma, K. and Buehler, M., editors (2006a). *Journal of Field Robotics - Special Issue on the DARPA Grand Challenge, Part 1*. Wiley Periodicals.

Iagnemma, K. and Buehler, M., editors (2006b). *Journal of Field Robotics - Special Issue on the DARPA Grand Challenge, Part 2*. Wiley Periodicals.

Jackel, L. D. (2005). Learning applied to ground robots (LAGR). http://www.darpa.mil/ipto/programs/lagr/.

Jackel, L. D., Krotkov, E., Perschbacher, M., Pippine, J., and Sullivan, C. (2006). The DARPA LAGR program: Goals, challenges, methodology, and phase I results. *Journal of Field Robotics*, 23(11-12):945–973.

Jain, A. K., Ratha, N. K., and Lakshmanan, S. (1997). Object detection using Gabor filters. *Pattern Recognition*, 30:295–309.

Jochem, T., Pomerleau, D., and Thorpe, C. (1995). Vision-based neural network road and intersection detection and traversal. In *Proc. of International Conference on Intelligent Robots and Systems (IROS)*, volume 03, pages 344–349. IEEE.

Jolliffe, T. (1986). Principal component analysis. *New York: Springer-Verlag*.

Kaye, K. and Bower, T. (1994). Learning and intermodal transfer of information in newborns. *Psychological Science*, 5:286–288.

Kelly, A. and Stentz, A. (1998). Stereo vision enhancements for low-cost outdoor autonomous vehicles. In *Proc. of Robotics and Applications (RA)*.

Kelly, A., Stentz, A., Amidi, O., Bode, M., Bradley, D., Diaz-Calderon, A., Happold, M., Herman, H., Mandelbaum, R., Pilarski, T., Rander, P., Thayer, S., Vallidis, N., and Warner, R. (2006). Toward reliable off road autonomous vehicles operating in challenging environments. *International Journal of Robotics Research*, 25:449.

Kim, D., Sun, J., Oh, S., Rehg, J., and Bobick, A. (2006). Traversibility classification using unsupervised on-line visual learning for outdoor robot navigation. In *Proc. of International Conference on Robotics and Automation (ICRA)*. IEEE.

Konolige, K., Agrawal, M., Bolles, R. C., Cowan, C., Fischler, M., and Gerkey, B. (2008). *Outdoor Mapping and Navigation using Stereo Vision*, pages 179–190. Springer Tracts in Advanced Robotics. Springer Berlin / Heidelberg.

Kriegman, D., Triendl, E., and Binford, T. (1989). Stereo vision and navigation in buildings for mobile robots. *Transactions on Robotics and Automation*, 5(6):792–803.

Krotkov, E., Fish, S., Jackel, L., McBride, W., Perschbacher, M., , and Pippine, J. (2006). The DAPRA PerceptOR evaluation experiments. *Autonomous Robots*.

Krotkov, E. and Hebert, M. (1995). Mapping and positioning for a prototype lunar rover. In *Proc. of International Conference on Robotics and Automation (ICRA)*, pages 2913–2919. IEEE.

Kumar, S. and Hebert, M. (2005). A hierarchical field framework for unified context-based classification. In *Proc. of Int'l. Conference of Computer Vision (ICCV)*. IEEE.

Lades, M., Vorbruggen, J., Buhmann, J., Lange, J., von der Malsburg, C., Wurtz, R., and Konen, W. (1993). Distortion-invariant object recognition in the dynamic link architecture. *IEEE Trans. Computers*, 42(3):300–311.

Lawrence, S., Giles, C. L., Tsoi, A. C., and Back, A. D. (1997). Face recognition: a convolutional neural-network approach. In *IEEE Transactions on Neural Networks*.

LeCun, Y. and Bengio, Y. (1995). Convolutional networks for images, speech, and time-series. In *The Handbook of Brain Theory and Neural Networks*. MIT Press.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

LeCun, Y., Huang, F., and Bottou, L. (2004). Learning methods for generic object recognition with invariance to pose and lighting. In *Proc. of Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 97–104.

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

Leib, D., Lookingbill, A., and Thrun, S. (2005). Adaptive road following using self-supervised learning and reverse optical flow. In *Proc. of Robotics: Science and Systems (RSS)*. MIT Press.

Leordeanu, M. and Collins, R. (2005). Unsupervised learning of object features from video

sequences. In *Proc. of Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE.

Longega, L., Panzieri, S., Pascucci, F., and Ulivi, G. (2003). Indoor robot navigation using log-polar local maps. In *International IFAC Symposium on Robot Control*.

Lowe, D. (1999). Object recognition from local scale-invariant features. In *Proc. of Int'l. Conference of Computer Vision (ICCV)*, pages 1150–1157. IEEE.

Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV)*, 60(2):91–110.

Lowe, D. G. (1995). Similarity metric learning for a variable kernel classifier. *Neural Computation*, 7:72–85.

Maimone, M. W., Leger, P. C., and Biesiadecki, J. J. (2007). Overview of the Mars Exploration Rovers' autonomous mobility and vision capabilities. In *Proc. of International Conference on Robotics and Automation (ICRA)*. IEEE.

Manduchi, R., Castano, A., Talukder, A., and Matthies, L. (2003). Obstacle detection and terrain classification for autonomous off-road navigation. *Autonomous Robot*, 18:81–102.

Martinez, A. (2002). Recognizing imprecisely localized, partially occluded and expression variant faces from a single sample per class. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 24(6):748–763.

Martinez, A. and Benavente, R. (1998). The AR face database. *CVC Technical Report*, 24. `http://rvl1.ecn.purdue.edu/ aleix/aleix_face_DB.html`.

Matthies, L., Gat, E., Harrison, R., Wilcox, B., Volpe, R., and Litwin, T. (1995). Mars microrover

navigation: Performance evaluation and enhancement. In *Proc. of International Conference on Intelligent Robots and Systems (IROS)*, volume 1, pages 433–440. IEEE.

Matthies, L., Maimone, M., Johnson, A., Cheng, Y., Willson, R., Villalpando, C., Goldberg, S., Huertas, A., Stein, A., and Angelova, A. (2007). Computer vision on Mars. *International Journal of Computer Vision*, 75(1):67–92.

Meltzoff, A. and Borton, R. (1979). Intermodal matching in human neonates. *Nature*, 282:403–404.

Michels, J., Saxena, A., and Ng, A. Y. (2005). High speed obstacle avoidance using monocular vision and reinforcement learning. In *Proc. of International Conference on Machine Learning (ICML)*, pages 593–600. ACM.

Mori, G., Belongie, S., and Malik, J. (2005). Efficient shape matching using shape contexts. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 27(11).

Murphy, P., Torralba, A., and Freeman, W. (2003). Using the forest to see the trees: a graphical model relating features, objects and scenes. In *Advances in Neural Information Processing Systems (NIPS)*. MIT Press.

Nabbe, B. and Hebert, M. (2003). Where and when to look - how to extend the myopic planning horizon. In *Proc. of International Conference on Intelligent Robots and Systems (IROS)*. IEEE.

Ng, A., Jordan, M., and Weiss, Y. (2002). On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems (NIPS)*, 14:849–856.

Novak, C. and Shafer, S. (1992). Anatomy of a color histogram. In *Proc. of Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE.

Oriolo, Ulivi, G., and Vendittelli, M. (1997). Fuzzy maps: A new tool for mobile robot perception and planning. In *Journal of Robotic Systems, 14(3), 179-197*.

Osadchy, M., LeCun, Y., and Miller, M. (2007). Synergistic face detection and pose estimation with energy-based models. *Machine Learning Research*, 8:1197–1215.

Pagnot, R. and Grandjean, P. (1995). Fast cross country navigation on fair terrains. In *Proc. of International Conference on Robotics and Automation (ICRA)*, pages 2593 –2598. IEEE.

Pantofaru, C., Dorku, G., Schmid, C., and Hebert, M. (2006). Combining regions and patches for object class localization. The Beyond Patches Workshop at Conference on Computer Vision and Pattern Recognition.

Pantofaru, C., Unnikrishnan, R., and Hebert, M. (2003). Toward generating labeled maps from color and range data for robot navigation. In *Proc. of International Conference on Intelligent Robots and Systems (IROS)*. IEEE.

Point Grey Research Inc. (2003). Triclops stereo vision software development kit user's guide and command reference. www.ptgrey.com/products/triclopsSDK.

Pomerleau, D. (1989). Alvinn: An autonomous land vehicle in a neural network. In *Advances in Neural Information Processing Systems (NIPS)*. Morgan Kaufmann.

Pomerleau, D. (1993). Knowledge based training of artificial neural networks for autonomous driving. *J. Connell and S. Mahadevan, eds., Robot Learning*.

Ranzato, M., Boureau, Y., Chopra, S., and LeCun, Y. (2007a). A unified energy-based framework for unsupervised learning. In *Proc. of Conference on Artificial Intelligence and Statistics (AI-STATS)*.

Ranzato, M., Boureau, Y., and LeCun, Y. (2007b). Sparse feature learning for deep belief networks. In *Advances in Neural Information Processing Systems (NIPS)*. MIT Press.

Ranzato, M., Huang, F., Boreau, Y., and LeCun, Y. (2007c). Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Proc. of Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE.

Rieder, A., Southall, B., Salgian, G., Mandelbaum, R., Herman, H., Render, P., and Stentz, T. (2002). Stereo perception on an off road vehicle. *Intelligent Vehicles*.

Riesenhuber, M. and Poggio, T. (1999). Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 2:1019–1025.

Ritt, J. T., Andermann, M. L., and Moore, C. I. (2008). Embodied information processing: Vibrissa mechanics and texture features shape micromotions in actively sensing rats. *Neuron*, pages 599–613.

Rizvi, S., Phillips, P., and Moon, H. (1998). The feret verification testing protocol for face recognition algorithms. *Technical Report NISTIR* 6,281, *Nat'l Inst. Standards and Technology*. `http://www.nist.gov/itl/div894/894.03/pubs.html#face`.

Rosenberg, C. and Hebert, M. (2002). Training object detection models with weakly labeled data. In *Proc. of the British Machine Vision Conference*.

Rosenberg, C., Hebert, M., and Schneiderman, H. (2005). Semi-supervised self-training of object detection models. In *Workshop on Applications of Computer Vision*. IEEE.

Roweis, S. and Saul, L. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500).

Schőelkopf, B., Smola, A., and Muller, K. (1998). Nonlinear component analysis as a kernel eigen-value problem. *Neural Computation*, 10:1299–1219.

Sermanet, P., Hadsell, R., Ben, J., Erkan, A. N., Flepp, B., Muller, U., and LeCun, Y. (2007). Speed-range dilemmas for vision-based navigation in unstructured terrain. In *Proc. of the IFAC Symposium on Intelligent Autonomous Vehicles (IAV)*. New York : Pergamon.

Sermanet, P., Hadsell, R., Scoffier, M., Muller, U., and LeCun, Y. (2008). Mapping and planning under uncertainty in mobile robots with long-range perception. In *Proc. of International Conference on Intelligent Robots and Systems (IROS)*. IEEE.

Shakhnarovich, G. and Moghaddam, B. (2004). Face recognition in subspaces. In Z. Li, S. and K. Jain, A., editors, *Handbook of Face Recognition*. Springer-Verlag.

Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, pages 888–905.

Simard, P., LeCun, Y., Denker, J., and Victorri, B. (2000). Transformation invariance in pattern recognition – tangent distance and tangent propagation. *International Journal of Imaging Systems and Technology*, 11(3).

Singh, S., Simmons, R., Smith, T., Stentz, A., Verma, V., Yahja, A., and Schwehr, K. (2000). Recent progress in local and global traversability for planetary rovers. In *Proc. of International Conference on Robotics and Automation (ICRA)*, pages 1194–1200. IEEE.

Sofman, B., Lin, E., Bagnell, J., Vandapel, N., and Stentz, A. (2006). Improving robot navigation through self-supervised online learning. In *Proc. of Robotics: Science and Systems (RSS)*. MIT Press.

Stavens, D. and Thrun, S. (2006). A self-supervised terrain roughness estimator for off-road autonomous driving. In *Proc. of Conference on Uncertainty in Artificial Intelligence (UAI)*.

Sudderth, E., Torralba, A., Freeman, W., and Willsky, A. (2005). Learning hierarchical models of scenes, objects, and parts. In *Proc. of Int'l. Conference of Computer Vision (ICCV)*. IEEE.

Sudderth, E., Torralba, A., Freeman, W., and Willsky, A. (2006). Describing visual scenes using transformed Dirichlet processes. In *Advances in Neural Information Processing Systems (NIPS)*. MIT Press.

Tenenbaum, J., DeSilva, V., and Langford, J. (2000). A global geometric framework for non linear dimensionality reduction. *Science*, 290:2319–2323.

Tesauro, G. (1992). Practical issues in temporal difference learning. *Machine Learning*, 8:257–277.

Thompson, B., Marks, R., and Choi, J. (2002). Implicit learning in autoencoder novelty assessment. In *Proc. of the International Joint Conference on Neural Networks*.

Thorpe, C., Hebert, M., Kanade, T., and Shafer, S. (1988). Vision and navigation for the Carnegie-Mellon Navlab. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 10(3):362–372.

Thrun, S. (1998). Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71.

Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G., Lau, K., Oakley, C., Palatucci, M., Pratt, V., and Stang, P.

(2006). Stanley: The robot that won the darpa grand challenge. *Journal of Field Robotics*, 23(9):661–692.

Torralba, A. and Oliva, A. (2002). Depth estimation from image structure. In *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*. IEEE.

Torralba, A. and Sinha, P. (2001). Statistical context priming for object detection. In *Proc. of Int'l. Conference of Computer Vision (ICCV)*. IEEE.

Torralba, A. B., Murphy, K. P., and Freeman, W. T. (2004). Contextual models for object detection using boosted random fields. In *Advances in Neural Information Processing Systems (NIPS)*. MIT Press.

Turk, M. and Pentland, A. (1991). Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1).

Ulrich, I. and Nourbakhsh, I. (2000). Appearance-based obstacle detection with monocular color vision. In *Proc. of Conference of the American Association for Artificial Intelligence (AAAI)*, pages 866–871.

Vandapel, N., Huber, D., Kapuria, A., and Hebert, M. (2004). Natural terrain classification using 3d ladar data. In *Proc. of International Conference on Robotics and Automation (ICRA)*. IEEE.

Vincent, P. and Bengio, Y. (2000). A neural support vector network architecture with adaptive kernels. *In Proc. of the International Joint Conference on Neural Networks*, 5.

Wagner, M., Baird, J., and Barbaresi, W. (1980). The locus of environmental attention. *Journal of Environmental Psychology*, 1:195–206.

Weinberger, K., Sha, F., and Saul, L. (2004). Learning a kernel matrix for nonlinear dimensionality reduction. *In Proceedings of the Twenty First International Conference on Machine Learning (ICML-04)*, pages 839–846.

Wellington, C. and Stentz, A. (2004). Online adaptive rough-terrain navigation in vegetation. In *Proc. of International Conference on Robotics and Automation (ICRA)*. IEEE.

Weston, J., Ratle, F., and Collobert, R. (2008). Deep learning via semi-supervised embedding. In *Proc. of International Conference on Machine Learning (ICML)*. ACM.

Winder, S. and Brown, M. (2007). Learning local image descriptors. In *Proc. of Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE.

Yagi, Y., Nagai, H., Yamazawa, K., and Yachida, M. (2001). Reactive visual navigation based on omnidirectional sensing - path following and collision avoidance. *Intelligent and Robotic Systems*, 31(4):379–395.

Zhang, H. and Ostrowski, J. (2002). Visual motion planning for mobile robots. In *In Proc. of Robotics and Automation, IEEE Transactions*.