Classifier Fusion for Outdoor Obstacle Detection

Cristian S. Dima, Nicolas Vandapel and Martial Hebert

Carnegie Mellon University
The Robotics Institute
Pittsburgh, PA 15217, USA
cdima,vandapel,hebert@ri.cmu.edu

***Abstract*— This paper describes an approach for using several levels of data fusion in the domain of autonomous off-road navigation. We are focusing on outdoor obstacle detection, and we present techniques that leverage on data fusion and machine learning for increasing the reliability of obstacle detection systems. We are combining color and infrared (IR) imagery with range information from a laser range finder. We show that in addition to fusing data at the pixel level, performing high level classifier fusion is beneficial in our domain. Our general approach is to use machine learning techniques for automatically deriving effective models of the classes of interest (obstacle and non-obstacle for example). We train classifiers on different subsets of the features we extract from our sensor suite and show how different classifier fusion schemes can be applied for obtaining a multiple classifier system that is more robust than any of the classifiers presented as input. We present experimental results we obtained on data collected with both the eXperimental Unmanned Vehicle (XUV) and a CMU developed robotic vehicle.**

I. Introduction

One of the most challenging aspects of autonomous navigation is perception in unstructured or weakly structured outdoor environments such as forests, small dirt roads and terrain covered by tall vegetation. In this paper, we focus on obstacle detection, where we consider an obstacle to be any region that a vehicle should not attempt to traverse (e.g. humans, trees, big rocks, large holes, large amounts of water).

We believe that in order to achieve acceptable levels of autonomy, vehicles operating in off-road conditions will need to rely on redundancies both at the sensor level and in the decision-making process. This redundancy should lead to better inferences about the scene observed and in turn to higher degrees of reliability of the obstacle detection system. Because each sensor is sensitive to different environmental conditions, the use of multiple of them allows the robot to deal with a larger variability of environment and operation conditions. In addition to data fusion, our approach relies quite heavily on machine learning. Detecting obstacles in environments that are as complex as the ones we are considering requires complex decision schemes which involve large numbers of parameters. Deriving such schemes manually is an extremely tedious process, if feasible at all, and leads to specialized systems. Instead we would like our robots to be easily adaptable to new environments and operating conditions, and for this purpose we use automated methods for tuning our systems.

The approach we propose in this context is illustrated in Figure 1: input data from multiple sensors is used by different classifiers whose results in turn are fused together to produce a unique classification result. We conceive it as a "black box" where machine learning techniques are used to automatically tune the classifiers and fuse their results in such way that the "black box" increases the likelihood of correct classification.

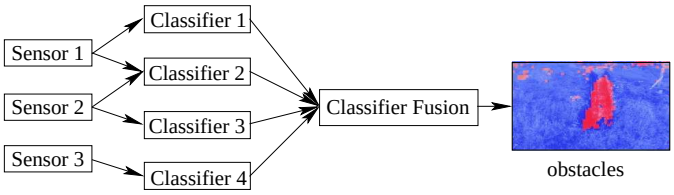


Fig. 1. Multiple classifiers operate on sensor data and are fused.The strategy is learned from training data

In this paper we present results based on several classifier combination techniques and show that such a black-box can be built in practice.

Using several sensing modalities or machine learning are certainly not new ideas in the mobile robotics field. A quick look at the previous work shows that sensor fusion has been a constant presence in this area from the earliest mobile robots with Hilare [1] to the platforms that define the current state of the art (ARL Demo III [2]). In 1992, Pomerleau [3] demonstrated the first successful application of machine learning methods to the problem of mobile robot navigation. It is interesting to contrast the machine learning techniques used in early robotic systems such as NAVLAB [4] to more recent approaches such as the Demo III project [5]. While the early systems tried to achieve autonomy by solving one monolithic learning problem (training a neural network to map from grey level images to steering angles in the case of Pomerleau's ALVINN [3]), more recently the trend has been to make intensive use of human domain knowledge and only use learning for those aspects of the problem that are hard to pre-program. For example, in [5] the authors describe a system which uses manually derived rules to identify geometric obstacles, and then filters the results through a color-based classifier that tries to identify the false geometric obstacles caused by vegetation. Figure 1 shows that our approach is located somewhere between the two extremes we just described.

In the following sections we describe the choice of the classification space (section II) and the fusion techniques we have experimented with (section III). In section IV we present

our experimental results and we draw conclusions and discuss future research directions in section V.

II. Choice of the classification space

Our approach assumes that the robot has 3-D and 2-D imaging capabilities through the use of a laser radar or stereo cameras, and through the use of color, black and white and/or infrared cameras. We assume also that the sensors calibration allows the 3-D data to be put into correspondence with the 2-D data. Finally, we assume that the robot has pose estimation to allow data integration. This last assumption can be relaxed without loss of generality for our approach.

The classification can be performed into either the 3-D space or the 2-D image space. Classification in the 3-D space requires a mapping of the features extracted from images to 3-D locations. Classification in the image space requires a mapping of the features extracted in the 3-D space to image patches and then the back-projection of the pixels classified as obstacle. Images are divided into a grid of rectangular regions we call patches. While the two representation models are essentially equivalent, we have chosen to use the image space classification which is more convenient for both labeling data and visualization of the classification results.

III. Classifier Fusion

A. Motivation

We assume that features are extracted from several different sensors. The features are described in Section IV-B. If we reduced ourselves to simply concatenating all the feature vectors we would essentially perform a simple form of data fusion at the pixel (or more precisely image patch) level. In many mobile robotics applications it is beneficial to be able to also include already existing classifiers that might incorporate significant amounts of domain knowledge. As we have stated in the introduction, we would like to have the capability to automatically learn when to use certain classifiers and how to combine them with and based on the available input data.

The reasons for which classifier combination might be desirable in robotics applications include:

- Different algorithms may be available, even using the same sensor, for terrain analysis. It is likely that the failure modes of their algorithms will be slightly different, which leads to the question whether by aggregating the decisions of all the classifiers in the pool we could do better on average than any individual algorithm.
- Certain types of obstacles can be particularly difficult to detect: thin wires and negative obstacles (such as holes and trenches) are good examples. While in such cases it might hard to implement a general obstacle detection algorithm that "learns" how to detect them, human understanding of the constraints specific to the obstacle to be detected can lead to much more effective *specialized* detectors. Learning classifier fusion automatically would enable us to determine the weights and rules that should be used with such specialized classifiers without manually tuning parameters based on their false alarms and detection rates.

B. Algorithms

In this paper we will discuss three algorithms for classifier combination: committees of experts [6], [7], stacked generalization [8] and a slight variation of the AdaBoost algorithm [9]. While our classifier fusion experiments are not limited to these specific algorithms, we consider them to be different enough from each other to be representative for the results one could expect from applying classifier fusion in our domain.

1) Committees of Experts: Initially described as a method for improving regression estimates in [10], [6], a committee of experts can be used for both regression and classification. The idea behind the algorithm is simple: if we have a pool of L experts that estimate a target function $f(x)$, we can linearly combine their outputs as $f_{COE}(x) = \sum_{i=1}^{L} \alpha_i f_i(x)$, where $f_i(x)$ is the estimate produced by the i^{th} expert. Under this model it can easily be shown [10], [6], [7] that the optimal (in the mean squared error sense) α_i's are given by

$$\alpha_i = \frac{\sum_{j=1}^{L} (\mathbf{C^{-1}})_{ij}}{\sum_{k=1}^{L} \sum_{j=i}^{L} (\mathbf{C^{-1}})_{kj}}$$

where $\mathbf{C}$ is the error correlation matrix. It can be shown that the mean squared error of the committee is always smaller than or equal to the average mean squared error over the classifier pool. In fact, if we assume that the experts make uncorrelated zero mean errors the error decreases by at least a factor of L. Obviously, this is overly optimistic: in reality the errors of the classifiers are going to be correlated so the reduction in error will be much smaller. However, given the simplicity of the method it is very attractive to use it. The assumption that needs to be made for the COE fusion approach is that the classifiers in the pool are trying to solve the same problem. As a result, this technique has the limitation of not being able to support specialized classifiers.

2) Stacked Generalization: Introduced by Wolpert in 1990 [8], stacked generalization (or "stacking") was initially presented as a method for combining multiple models learned for classification. Since then, stacking has also been used for regression [11] and even unsupervised learning [12].

Despite being an extremely simple algorithm, stacked generalization is quite difficult to describe. To make the task easier, we describe what stacked generalization (SG) would be equivalent to if we are willing to assume that a very large amount of training data is available, and then explain the actual algorithm.

In the form described by Wolpert in [8], stacked generalization is a two stage classifier. Just like in the case of committees of experts we will assume that we have a pool of L trainable experts that estimate a target function $f(x)$. These classifiers are what Wolpert calls the "level-0 generalizers", and are trained in the first stage of SG. The second stage consists of training a classifier that takes as inputs the outputs of the level-0 generalizers and tries to produce the correct label

as an output. This classifier is called the "level-1 generalizer", and its purpose is to learn the biases of the level-0 generalizers.

The crucial element of stacked generalization is that the level-1 generalizer should be trained using data that is "new" to the level-0 generalizers, since we are interested in learning about their generalization properties and not their ability to overfit. In the ideal case where very large amounts of training data were available, this could simply be achieved by splitting the training data and reserving half of it (for example) for training the second stage classifier. The only difference about the stacked generalization algorithm and the method we just described is that in the real algorithm a cross-validation scheme is used so that all the data is used for training both stages of the classifier.

Stacked generalization works surprisingly well in practice, and it has been applied successfully in other domains such as Automatic Target Recognition (ATR) [13].

3) AdaBoost with Classifier Selection: AdaBoost is an algorithm that has been shown to be somewhat similar to the popular support vector machines, in that it tries to maximize the separation margin. Shapire and Freund [9] proposed a clever iterative algorithm that solves the margin maximization problem with the only requirement that a so-called "weak classifier" –a learning algorithm that can perform better than a random one– is available.

The intuitive idea behind AdaBoost is to train a series of classifiers and to iteratively focus on the hard training examples. The algorithm relies on continuously changing the weights of its training examples so that those that are frequently misclassified get higher and higher weights: this way, new classifiers that are added to the ensemble are more likely to classify those hard examples correctly. Aside from this intuition, AdaBoost's training scheme corresponds to performing gradient descent on an error function that exponentially penalizes small classification margins [14], [15].

Our small variation to the regular form of Adaboost consists in allowing the algorithm to choose at each iteration which *type* of weak classifier to train. Assuming that we have a pool of classifiers and that some of them can be trained, we allow the algorithm to examine all the classifiers in our pool – training the ones that are trainable– and select the one that can best classify the training examples given their current weight distribution. Thus, AdaBoost will select one of the classifiers available at each iteration.

Note that while this is not the regular procedure for training AdaBoost, we are not modifying any of the assumptions that the algorithm is based on. A similar application of AdaBoost was successfully demonstrated by Tieu and Viola [16] in the context of automated image retrieval.

IV. Experimental Results

In order to validate the techniques described so far we performed experiments with two different vehicles equiped with two different sensor suites: the GDRS XUV and a CMU robotic platform.

A. Sensors registration

The XUV vehicle is equipped with the GDRS mobility ladar, two 640x480 pixels Sony color cameras and an infrared camera with the same resolution. The laser unit and the cameras are mounted inside a pan-tilt head. Figure 2 shows the XUV autonomous unmanned vehicle and a close view of the pan-tilt head. The laser is located in central part of head (the large rectangular optics) and the cameras are located on each side (smaller circular apertures). Information on the ladar can be found in [17].

The CMU developed robotic platform, a large tractor, is equipped with two Sony DFW-SX900 high-resolution color digital cameras producing 1280x960 pixels images and two laser range finder units which are based on mechanically scanned SICK LMS units. At the time the data logs were recorded the vehicle did not have an IR camera.

Fusing multi-sensor data at low-level requires solving the data association problem, which consists of establishing correspondences between the measurements returned by the different sensors. In our case we will need to find such correspondences between our laser data and the images from the color and IR cameras as we will show in section IV. The procedure is decomposed into two steps: 1) determining the intrinsic parameters of our color and IR cameras and 2) recovering the 3-D rigid transformation between the ladar reference frame and each cameras reference frame. In the first step we use the Matlab Camera Calibration Toolbox [18]. In the step second we build correspondances between the corners of a checkerboard calibration target in both the 3-D data and images. Thus each range measurement can be associated with color and IR information and vis-versa.

B. Features

For each patch in our image grids we compute five features: color, texture, infrared, laser (simple statistics) and laser(VH features). The details of the computation are described below.

1) Color: The images are converted to the LUV color space; we extract the mean and standard deviation of the pixels in a patch for each channel, obtaining 6 color features.

2) Texture: The Fast Fourrier Transform (FFT) representation of each patch is computed, and it is then divided into 6 bins for frequency and 6 for the orientation. The means and standard deviation of the energy in each bin are computed, resulting in a total of 24 texture features.

3) Infrared: The mean and standard deviation of the IR pixel values for each patch are computed, resulting in 2 IR features. The correspondence between the color patches (used as reference) and IR patches is established using the 3-D information provided by the laser points that project in the color patch.

4) Laser (simple statistics): Using the laser points that project into each image patch we estimate the average height expressed in the vehicle frame, and the standard deviations in the XYZ directions relative to the vehicle frame. This results in 4 simple laser features.

5) Laser (VH features): As a good example of a specialized classifier we might want to incorporate into our system, we have used an implementation of the technique described in [19] for terrain classification. The method looks at the local point distribution in space and uses a Bayes classifier to produce the probability of belonging to 3 classes - vegetation, solid surface and linear structure. The method takes as input a sparse set of 3-D points. At each point the scatter matrix is computed using a predefined support region. The decomposition in principal components of this matrix leads to the definition of three saliency features characterizing the 3-D points spatial distribution as ”random”, ”linear” and ”surface”. We use both these saliencies and the probabilities of belonging to each class, which results in a number of 6 features. We will refer to these features as “Laser VH”.

C. Experimentation with the XUV

The first experiment we present is based on data collected with the XUV robotic platform. We evaluated the performance of the various feature sets and the benefit of the different fusion strategies by attempting to solve a problem that is important for outdoor mobile robotics: detecting dirt roads. While the road detection is not an instance of an obstacle detection problem, notice that our setup is essentially solving binary classification problems and as such can also be used for terrain classification.

(a) XUV during a traverse

(b) Sensor suite on the turret

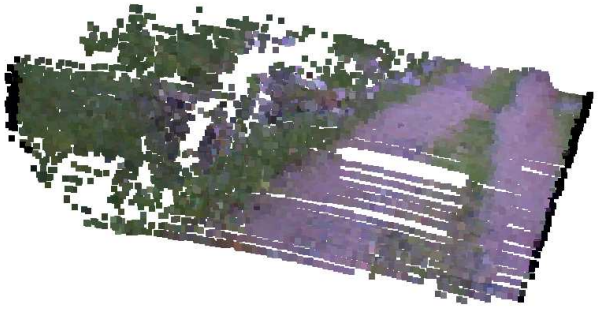
Fig. 2. XUV vehicle and GDRS ladar

The data logs used for this experiment were collected at the ARL Fort Indiantown Gap robotics facility. Each data log contained color and infrared images, together with vehicle position and range data from the vehicle. We have used 3 independent datasets (2 merged into the training set, 1 used as a test set). The corresponding images were manually labeled in the two classes of interest. We have only used image patches that contained laser points, which resulted in 18963/8582 patches in the train/test datasets. The percentage of road patches was 0.62/0.63.

After labeling the data and extracting the features we have trained several classifiers on this problem. More specifically, we compared the performance of neural networks trained on subsets of our full feature vector (such as color, texture, IR, laser simple and laser VH) with the performance of a neural network that has access to the full vector. We also compared their performance to two of our classifier fusion algorithms, stacked generalization and committees of experts. Table I

(a) Color image

(b) IR image



(c) 3-D point cloud in which points are colorized based on the color image

Fig. 3. A typical scene from the road detection dataset

presents the error rates for the road detection experiments. From the first row down we have stacked generalization, committees of experts, and color, texture, infrared, laser simple, laser VH, and all feature based neural networks.

TABLE I

ERROR RATES FOR THE ROAD DETECTION EXPERIMENTS.

Name	Mean	Std Dev
SG	2.89	0.44
CoE	3.77	0.54
Color	9.45	2.79
Texture	28.73	2.02
IR	12.33	5.22
Laser Simple	17.33	5.29
Laser VH	11.72	3.13
All Features	3.19	0.61

In order to estimate the error rates and standard deviations we performed 10 fold cross-validation without prior randomization of the patches. We chose not to use randomization in order to avoid getting overly optimistic results: since there is a high degree of correlation between neighboring image patches, splitting them randomly would lead to unrealistic similarities between the training and testing datasets. We have also performed experiments with completely separate training and test datasets (i.e. without cross-validation) and the error rates we obtained were similar to the ones produced by cross-validation.

Overall our results are encouraging: they confirm that performing both low-level data fusion and classifier fusion can significantly improve classification performance. The fact that committees of experts and stacked generalization performed as well as a neural network that has access to the full feature vector is very positive. While in this case we had full access

to all the features (including the ones produced by the VH classifier) which reduces the importance of classifier fusion, it is important to confirm that algorithms like COE and SG can learn to combine input classifiers very effectively.

It is interesting to notice that the VH features (which effectively represent a form of specialized classifier) perform significantly better than the simple laser statistics, despite the fact that exactly the same laser points are used as inputs in both cases. This is a perfect example of why one would like to be able to fuse several classifiers.

D. Experimentations with the CMU vehicle

The second experiment uses data collected with the CMU vehicle and the same types of features as the ones based on XUV data, except for the laser VH and the IR features, which were not available. The cameras and the laser units have performance characteristics that are quite different from those of the XUV sensors. We show that using automated learning makes our fusion techniques applicable to many different vehicles and sensor configurations. We present results on two instances of the obstacle detection problem: human detection and negative obstacle detection.

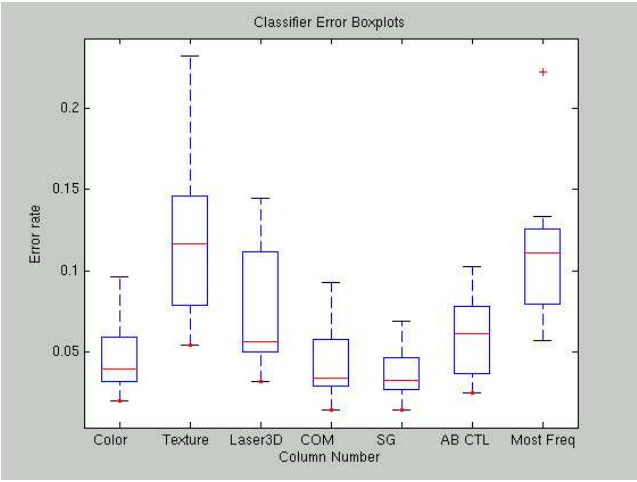


Fig. 4. Box plots representing the classification performance on the obstacle detection problem. The rectangle for each classifier represents the interquartile range and the horizontal line is the median. From left to right we have the color, texture and laser based classifiers, the committee of experts (COM), stacked generalization (SG), AdaBoost (AB CTL) and Most Frequent, a classifier that always predicts the most frequent class without using any features.

1) Human detection: In the dataset, the human obstacle was walking in front of the moving vehicle in an area with tall vegetation. To make the problem non-trivial the human was wearing a camouflage jacket. The raw classifiers were neural networks, this time using color, texture and simple laser features. The classifier fusion strategies we compared were stacked generalization, a committee of experts and the version of AdaBoost we described. The dataset we used contained 22989 non-obstacle and 2893 obstacle image patches (we used 20x20 patches).

The results presented in Figure 4 were obtained performing 10 fold cross-validation on our dataset. Since the two classes (obstacle/non-obstacle) were so unbalanced, we presented the error rate of a "constant" classifier that always predicts the most frequent class. Since only 12 percent of our data represents the obstacle class the reader should be aware that an error rate of 10 percent does not necessarily represent good performance.

In this experiment the color classifier performed extremely well, followed by the laser features and the texture which was mostly irrelevant. The explanation is that the vegetation was slightly dry, which made the color of the camouflaged jacket different from the background. Stacked generalization and the committee of experts were able to learn to focus on the color-based predictions and to use the laser information to slightly improve upon the color performance. A t-test based on our cross-validation data showed this slight improvement to be statistically significant.

The boosting algorithm performed slightly worse than the best input classifier. Our analysis indicated that the problem lies in the exponential penalty that AdaBoost "charges" for small classification margins. The algorithm focuses on increasing the margin on a small number of very difficult training examples while actually reducing the margin of the others; as a result, its generalization performance is reduced. A solution to this problem would be to use "soft-margin" AdaBoost variations such as the one described in [20].

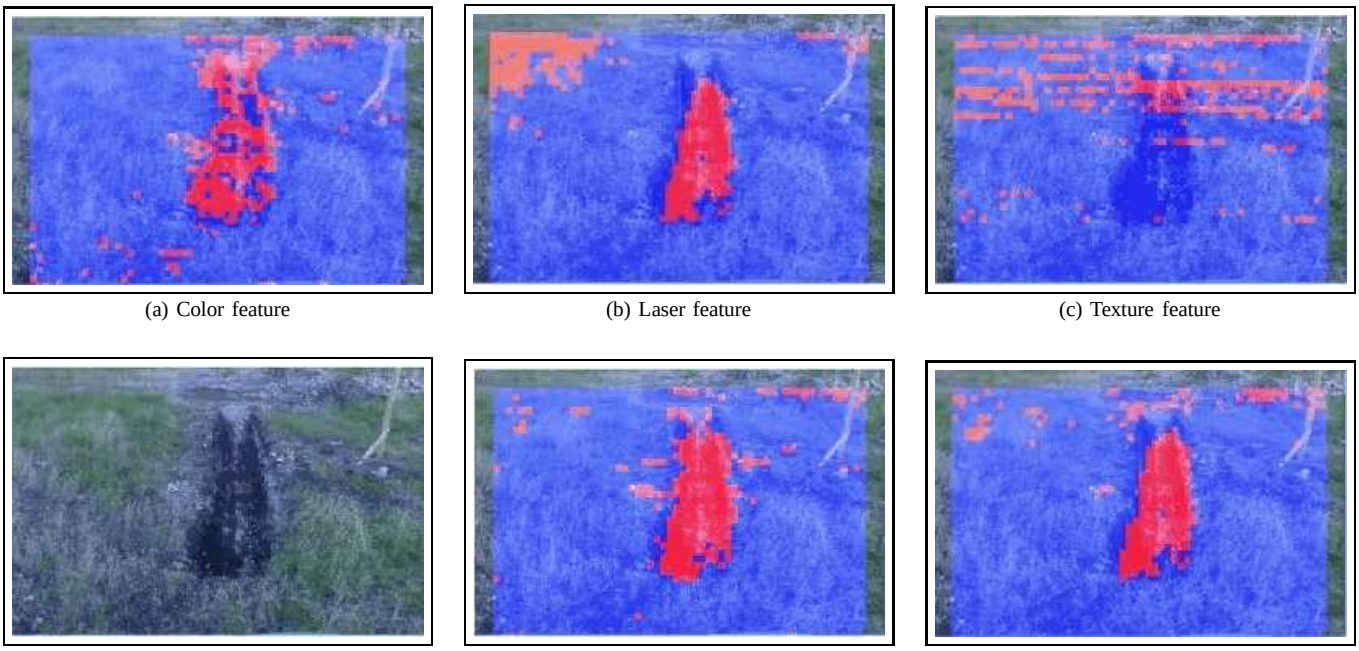
2) Negative obstacle detection: Figure 5 shows a result of negative obstacle detection by classifier fusion. The obstacle is large (0.75 m x 3 m) and deep (one meter) rectangular depression in the ground, locate at 3 meters from the vehicle. Data collected by two sensors (color camera and laser) were used by three classifiers (color, texture, and 3-D statistics). The classifiers result were fused using two methods, fusion neural network and stack generalization, to produce respectively the result image Figure 5-(e) and (f).

E. Implementation

The concept of "black box" for classifier fusion presented in this paper, can also be found in our software implementation. From the beginning, attention has been given to the portability (migration to another robot) and the versatility (use of new classifier) of the code.

V. Summary and Future Work

We have presented a system that uses multisensor data fusion at both the pixel level and the classifier level in order to improve obstacle detection performance for outdoor mobile robots. Our experiments –on different platforms, sensors and feature configurations– confirm the intuition that combining data from multiple sensing modalities can dramatically improve classification performance. Furthermore, we have shown that automatically combining different classifiers in order to leverage on their particular strengths and provide performance that is better than that of any classifier in the pool is possible. We will continue our experiments in order to analyze the



(a) Color feature (b) Laser feature (c) Texture feature

(d) Scene (e) Fusion NN (f) Fusion Gen. Stacked

Fig. 5. Example of obstacle detection with the CMU vehicle. The blue and red areas correspond respectively to common sensor field of view and obstacles.

performance of our system on different classification problems and with more complex classifier combination schemes such as hierarchical mixtures of experts [21]. The weakest link of our current setup is the fact that we rely on supervised learning. Labeling data for large scale problems is tedious and expensive, and we are currently developing active learning solutions for alleviating the data labeling requirements.

VI. Acknowledgements

We would like to acknowledge the valuable support of Carl Wellington in developing some of the infrastructure used for these experiments.

This paper was prepared through collaborative participation in the Robotics Consortium sponsored by the U. S. Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement DAAD19-01-2-0012. The U. S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon.

References

[1] A. de Saint Vincent, "A 3-D perception system for the mobile robot HILAIRE," in *IEEE International Conference on Robotics and Automation*.

[2] C. Shoemaker and J.Bornstein, "The Demo III UGV program: A testbed for autonomous navigation research," in *Proceedings of the 1998 IEEE ISIC/CIRA/ISAS Joint Conference*.

[3] D. Pomerleau, "Progress in neural network-based vision for autonoumous robot driving," in *IEEE International Symposium on Intelligent Vehicles*, 1992.

[4] S. Shafer, A. Stentz, and C. Thorpe, "An architecture for sensory fusion in a mobile robot," in *IEEE International Conference on Robotics and Automation*.

[5] P. Belluta, R. Manduchi, L. Matthies, K. Owens, and A. Rankin, "Terrain perception for DEMO III," in *IEEE International Symposium on Intelligent Vehicles*, October 2000.

[6] M. Perrone, "Improving regression estimation: Averaging methods for variance reduction with extensions to general convex measure optimization," Ph.D. dissertation, Brown University, 1993.

[7] C. Bishop, *Neural Networks for Pattern Recognition*. Oxford University Press, 1997.

[8] D. H. Wolpert, "Stacked generalization," Los Alamos, NM, Tech. Rep. LA-UR-90-3460, 1990.

[9] R. E. Schapire, "A brief introduction to boosting," in *International Joint Conference on Artificial Intelligence*, 1999.

[10] M. P. Perrone and L. N. Cooper, "When networks disagree: Ensemble methods for hybrid neural networks," in *Neural Networks for Speech and Image Processing*, R. J. Mammone, Ed. Chapman-Hall, 1993, pp. 126–142.

[11] L. Breiman, "Stacked regressions," *Machine Learning*, vol. 24, no. 1, 1996.

[12] P. Smyth and D. Wolpert, "An evaluation of linerly combining density estimators via stacking," Information and Computer Science Department, University of California, Irvine, Tech. Rep., 1998.

[13] L.-C. Wang, L. Chan, N. Nasrabadi, and S. Der, "Combination of two learning algorithms for automatic target recognition," in *IEEE Inernational Conference on Image Processing*, 1997.

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

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

[16] K. Tieu and P. Viola, "Boosting image retrieval," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2000.

[17] M. Shneier, T. Chang, T. Hong, G. Cheok, H. Scott, S. Legowik, and A. Lytle, "A repository of sensor data for autonomous driving research," in *Proceedings of the SPIE Aerosense Conference*, 2003.

[18] J.-Y. Bouguet, "Camera calibration toolbox for matlab," http://www.vision.caltech.edu/bouguetj/calib_doc/.

[19] M. Hebert and N. Vandapel, "Terrain classification techniques from ladar data for autonomous navigation," in *Collaborative Technology Alliance Workshop*, 2003.

[20] G. Rätsch, T. Onoda, and K.-R. Müller, "Soft margins for AdaBoost," *Machine Learning*, vol. 42, no. 3, 2001.

[21] M. Jordan and R. Jacobs, "Hierarchical mixtures of experts and the em algorithm," *Neural Computation*, no. 6, 1994.