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 tractor.**

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 reliability in obstacle detection, vehicles operating in off-road conditions will need to rely on multiple sensing modalities (such as color, infrared images or range measuments) and on several detection algorithms. Obstacle detection is an inference problem, because no sensor can directly measure "obstacleness": one needs to infer such information from measurements of color, temperature and shape. It is natural to expect that having more such sources of information can lead to better inferences and subsequently to more reliable navigation. Because each sensor is sensitive to different environmental conditions and has different failure modes, using multiple sensors on robotic vehicles will extend the range of conditions in which they can operate.

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 decision schemes which involve large numbers of parameters. Deriving such schemes manually is an extremely tedious process which leads to highly specialized systems that are hard to adapt to new environments and operating conditions. Using machine learning to automatically tune our system enables us to avoid this significant problem.

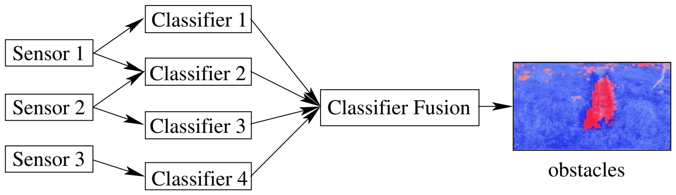


Fig. 1. Multiple classifiers operate on sensor data and their results are fused to detect obstacles. The fusion strategy is learned from training data.

The approach we propose is illustrated in Figure 1. Input data from multiple sensors is used by different classifiers whose results are fused to produce a unique classification result. We would like our classifier fusion step to function as a "black box" where learning techniques are used to automatically evaluate or even tune the classifiers and combine their results. We expect the black box to output correct classification more often than any of the input classifiers. 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 multiple 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 [2]–[4]. In 1992, Pomerleau [5] 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 ALVINN [5] to more recent approaches such as ones used in the Demo III [6] or the PerceptOR [4] programs. 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 [5]), 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. In [6] the authors describe a system which uses manually derived rules to identify geometric obstacles, and then filters the results through a trainable color-based classifier meant to identify the false geometric obstacles caused by vegetation. Similarly, Stentz et al. [4] describe a system that uses a neural network to estimate compressibility from color data and then combines it through manually designed rules with other sources of information.

Our approach is located somewhere between the two extremes we just described: we would like to be able to use human domain knowledge when it is available, but we want to avoid having to manually derive classifier combination rules. Our main goal is to minimize the number of parameters that need to be manually adjusted while maintaining high performance levels.

In the following section we motivate our interest in classifier fusion and we describe the algorithms we have experimented with. In Section III we present our experimental setup and some our results. Finally, we draw conclusions and discuss future research directions in Section IV.

II. Classifier Fusion

A. Motivation

Let us assume that we are interested in classifying a location in front of a robot as being traversible or not. The sensors mounted on the vehicle can provide various measurements about it: we could estimate its average temperature, color, and through some image processing algorithms we could extract texture information from our images. Once all these features are available, we need a method to combine them in order to classify the location in one of the two possible classes.

If we reduced ourselves to simply concatenating all the features we would essentially perform a simple form of data fusion at the pixel level. While pixel level fusion is a verified and valuable method for combining data from multiple sensors, in mobile robotics applications there are benefits that come from also being able to fuse information at a higher level.

We have mentioned earlier that many of current state-of-the-art robotic systems incorporate a significant amount problem specific human knowledge. For example, some of the algorithms used in the PerceptOR program [4] have been extensively tuned to perform well in vegetated forest areas. Even if our main approach is to use machine learning for obstacle detection, it would be unfortunate to discard such valuable previous knowledge and try to learn everything directly from raw sensor data. The automatic classifier fusion approach we introduced in Section I would be capable of taking several such specialized classifiers as input (possibly along with some raw sensor data) and learning from training data how well they perform and how their outputs should be combined.

Most of the algorithms that map from sensor data to high level classifications make different assumptions about the environment and about the sensors. These different assumptions will usually ensure that the classification errors of the various algorithms are not perfectly correlated. There is then a potential of pooling their predictions together and obtaining classification results that are better than each individual classifier.

Finally, there are certain types of obstacles that are extremely hard to detect. Thin wires and negative obstacles (holes and trenches) have small signatures that make it challenging to learn how to detect them directly from sensor data. However, the human understanding of the nature of these obstacles can lead to effective detectors. We know for example that the 3-D laser points returned from a wire will form an alongated point cloud whose scatter matrix will have one very large and two small eigenvalues. The same wire will generate an edge in our images, and we know that the Canny edge detector can be used to detect it. By correlating this type of information we can obtain much better results than if we tried to learn everything from sensor data. We cannot realistically expect current algorithms to "learn" the details of eigen-analysis and edge detection from the amounts of training data we have available. On the other hand, developing specialized detectors for all the possible obstacles in off-road environments is equally unrealistic. Once again, we believe that the right solution is being able to automatically learn how to fuse both pre-programmed specialized detectors and classifiers that use learning.

B. Algorithms

In this paper we will discuss three algorithms that can be used for classifier combination: committees of experts [7], [8], stacked generalization [9] and a slight variation of the AdaBoost algorithm [10].

1) Committees of Experts: Initially described as a method for improving regression estimates in [7], [11], a committee of experts can be used for both regression and classification.

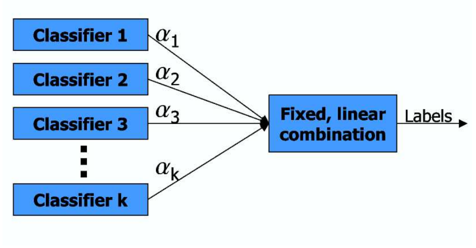


Fig. 2. Committee of Experts: the output of the committee is a fixed linear combination of the input classifiers.

The idea behind the algorithm is simple: if we have a pool of L experts that estimate a target function $h(x)$, 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. If we express the estimation error of each one of the experts as $\epsilon_i(x) = f_i(x) - h(x)$ it can easily be shown [8] that the optimal α_i's in the mean squared error sense 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 with

$$C_{ij} = E[\epsilon_i(x)\epsilon_j(x)]$$

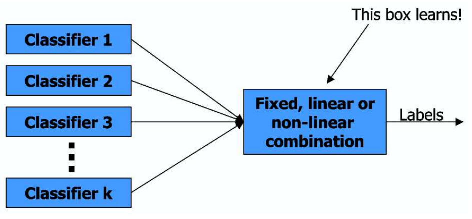


Fig. 3. Stacked Generalization: the level-1 learner is trained to compensate the biases of the level-0 learners.

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.

This method assumes that the classifiers in the pool are trying to solve the same classification problem. As a result, committees of experts are only adequate for fusing classifiers that attempt to detect the entire set of the obstacles of interest.

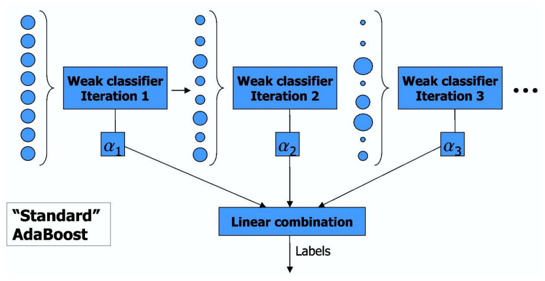
2) Stacked Generalization: Introduced by Wolpert in 1990 [9], stacked generalization (or "stacking") was initially presented as a method for combining multiple models learned for classification. Since then, stacking has also been used for regression [12] and even unsupervised learning [13].

In the form described by Wolpert in [9], stacked generalization (SG) 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 $h(x)$. 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 level-1 generalizer should be trained using data that is new to the level-0 generalizers. We are interested in learning about their generalization properties and not their ability to overfit. Without new data it would be impossible for the level-1 learner to distinguish between a level-0 generalizer that has very good generalization properties and one that simply memorized the training data perfectly.

In the ideal case where very large amounts of training data were available, obtaining new data for training the level-1 learner could be achieved by splitting the training data and reserving half of it (for example) for training the second stage classifier. The defining detail about stacked generalization consists in a cross-validation-like scheme that makes it possible to use all the data for training both stages of the classifier while still avoiding overfitting problems.

Stacked generalization works very well in practice, and it has been applied successfully in other domains such as Automatic Target Recognition (ATR) [14].



Weak classifier Iteration 1
Weak classifier Iteration 2
Weak classifier Iteration 3
α_1
α_2
α_3
"Standard" AdaBoost
Linear combination
Labels

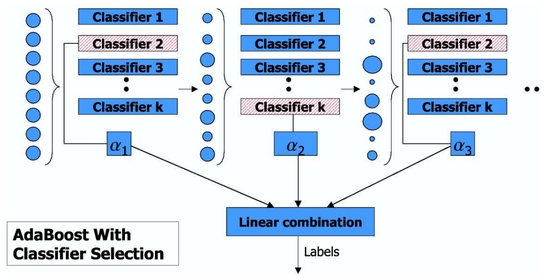


Fig. 4. AdaBoost. TOP: the standard form of AdaBoost. The varying diameters of the blue (dark) circles represent the changing weights over the training examples. At each iteration, the weak classifer is trained on the current distribution over the training data. BOTTOM: our modification of AdaBoost. Insted of considering only one form of weak classifier, we train all the classifiers in our pool and select the best performing one at each iteration.

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 between two classes. Shapire and Freund [10] 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. As shown in Figure 4, 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. In the end, AdaBoost predicts one of the classes based on the sign of a linear combination of the weak classifiers trained at each step. The algorithm generates the coefficients that need to be used in this linear combination.

Shortly after its publication, AdaBoost raised a lot of interest when several experiments have shown that it seemed not to overfit the training data even as thousands of weak classifiers were added to the ensemble. Since then it has been shown that AdaBoost's training scheme corresponds to performing gradient descent on an error function that exponentially penalizes small classification margins [15], [16]. AdaBoost can and does overfit, especially in noisy domains.

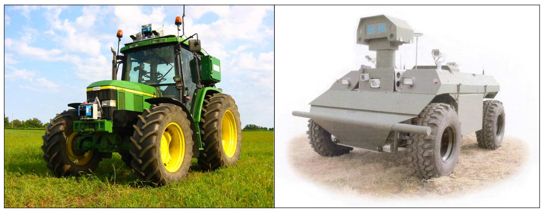


Fig. 5. The CMU robotic tractor (left) and the XUV (right)

It is however still a widely used algorithm due to its ability to “boost” the performance of very weak classifiers.

Our variation to the regular form of Adaboost consists in allowing the algorithm to choose at each iteration among several *types* of weak classifiers. 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.

Notice 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. Allowing AdaBoost to use several types of weak classifiers is equivalent to having single weak classifier algorithm with an extended hypothesis space. As a result, all the convergence proofs that apply to AdaBoost also apply to our version of the algorithm.

A similar application of AdaBoost was successfully demonstrated by Tieu and Viola [17] in the context of automated image retrieval.

III. Experimental Results

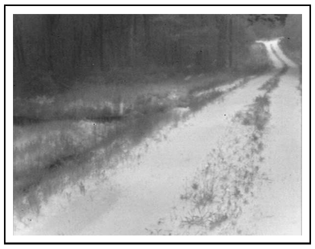
In order to validate the techniques described so far we performed experiments with data coming from two different robotic vehicles: an eXperimental Unmanned Vehicle (XUV) developed by General Dynamics and a CMU-developed robotic tractor.

A. Sensors and classification space

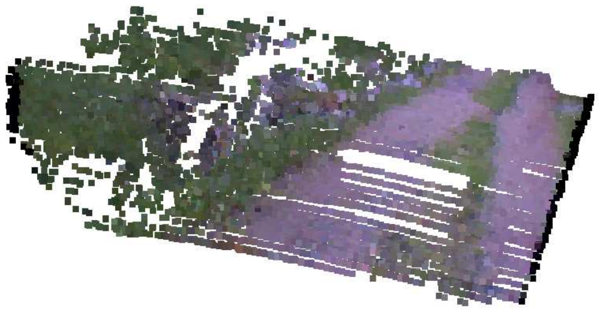
The XUV vehicle (see Figure 5) is equipped with a laser range finder, two color cameras and an infrared camera. Information on the sensors can be found in [18], [19]. The CMU developed robotic tractor is equipped with two Sony DFW-SX900 high-resolution color digital cameras producing 1280x960 pixels images and two laser range finder units which are based on mechanically scanned SICK LMS-200 units. At the time the data logs used in this paper were recorded the vehicle did not have an infrared camera.

Fusing multisensor data at low level requires solving the data association problem, which consists of establishing correspondences between the measurements returned by the different sensors. We have developed a calibration procedure that enables us to precisely map 3-D points sensed by the laser range finders to our images. As a result we can generate “colorized” three-dimensional maps such as the one shown in Figure 6(c), and we can obtain an approximate 3-D location for each small patch from our images.

(a) Color image



(b) IR image



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

Fig. 6. A typical scene from the road detection dataset

We have chosed to use the image space extended with 3-D information for solving our classification problems. One of the color images is divided in a rectangular grid of patches, and the 3-D points returned by the laser range finder are mapped to their corresponding patches. Features based on various sensing modalities are generated for each image patch, and they are used for classification. The classification results can be projected back in the 3-D world using the laser range information residing our grid.

B. Features

For each patch in our image grid we compute five types of features: color, texture, infrared, and two types of laser features.

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 six color features.

2) Texture: We use a Fast Fourrier Transform (FFT) representation of each patch in order to measure the amount of energy present at different orientations and frequencies. For the experiments described in this paper we extracted twenty-four texture features for each image patch.

3) Infrared: The mean and standard deviation of the IR pixel values for each patch are computed, resulting in two 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 forward, lateral and upward 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 want to incorporate in our system, we have used an implementation of the technique described in [20] for terrain classification. The method takes as input a sparse set of 3-D points and looks at the local point distribution in space. Based on the local points a scatter matrix is computed and its eigenvalues are used to obtain three saliency features describing the 3-D point distribution as "random", "linear" or "surface". A Bayes classifier using these three features is finally used to estimate the probability of a point of belonging to one of the three classes mentioned above. We will refer to the three probabilities and the three saliencies as the "Laser VH" features.

C. Experiments with the XUV

The first experiment we present is based on data collected with the XUV robotic platform. We evaluate the performance of the various feature sets and the benefit of the different fusion strategies on an important problem for outdoor mobile robotics: detecting dirt roads. While the road detection is not an instance of an obstacle detection problem, our setup is essentially solving binary classification problems and as such can also be used for two-class terrain classification with absolutely no changes.

The data logs used for this experiment were collected at a test site in central Pennsylvania. Each data log contained color and infrared images, together with vehicle position and range data. We have used 3 independent datasets (2 merged into the training set, 1 used as an independent test set). The corresponding images were manually labeled in the two classes of interest. The train set contained 18963 patches(62% "road") and the test set contained 8582 patches (63% "road").

The data was manually labeled and we have trained several classifiers on this problem. At the time of these experiments we did not have truly specialized or hand-tuned road detectors, so we have trained neural networks on subsets of our full feature vector (such as color, texture, IR, laser simple and laser VH) to obtain classifiers based on the different types of information. We compared their performance to a neural network that has access to the full feature vector (the maximum amount of information). We also compared their performance to two of our classifier fusion algorithms, stacked generalization and committees of experts. Table I presents the error rates for the road detection experiments. From the first row down we have stacked generalization, a committee of experts, and color, texture, infrared, laser simple, laser VH, and all feature based neural networks.

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 involuntary contamination 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.

TABLE I
ERROR RATES FOR THE ROAD DETECTION EXPERIMENTS

Name	Mean	Std Dev
SG	2.89	0.44
CoE	3.77	0.54
All Features	3.19	0.61
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

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 much better then the individual classifiers they took as input and as well as the neural network that has access to the full feature vector is very positive.

It is interesting to notice that the VH features perform significantly better than the simple laser statistics, despite the fact that precisely 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 use specialized classifiers: the VH features encode human knowledge about the 3-D statistics of point clouds coming from flat surfaces, and this additional information leads to better performance on the road detection task.

The error of the texture based network is by far the largest, being very close to random predictions. After performing these experiments we have discovered that some of the settings we used for texture feature extraction were accidentally set to the wrong values. This somewhat unfortunate event helped however prove a very important point: the two classifier fusion algorithms we tested here are robust to having extremely weak classifiers in their pool. They both learned to largely ignore the texture prediction and only use the color, IR and laser features instead.

D. Experiments 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 sensors on the robotic tractor have performance characteristics that are quite different from those of the XUV sensors. Being able to switch our entire obstacle detection system between two different vehicles by simply changing the training datasets shows the power that comes with the use of automated learning techniques. We expect our methods to be easily transferable to many other robotic platforms.

In this section we present results on two instances of the obstacle detection problem: human and negative obstacle detection.

1) Human detection: In this dataset a human walks in front of the moving vehicle in an area with tall vegetation. To make

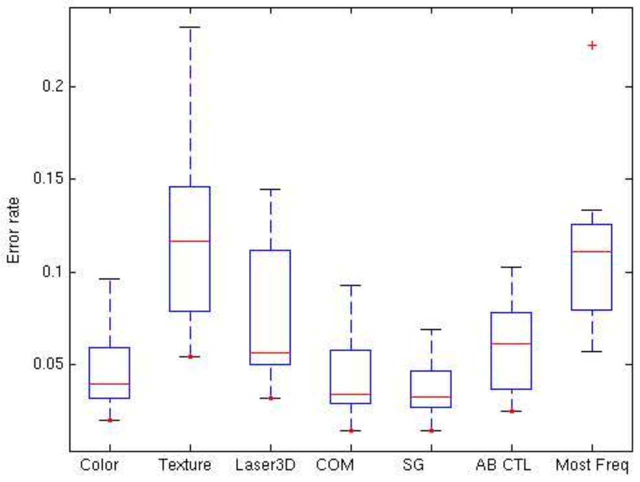


Fig. 7. Box plots representing the classification performance on the obstacle detection problem. The rectangle for each classifier represents the interquartile range and the horizontal line is the median. From left to right we have the color, texture and laser based classifiers, the committee of experts (COM), stacked generalization (SG), AdaBoost (AB CTL) and Most Frequent, a classifier that always predicts the most frequent class without using any features.

the problem non-trivial the human is wearing a camouflage jacket. The "specialized" classifiers are again neural networks, this time trained using color, texture and simple laser features. The classifier fusion strategies we compare are stacked generalization, a committee of experts and the version of AdaBoost we described. The dataset contains 22989 non-obstacle and 2893 obstacle image patches.

The results we present in Figure 7 were obtained performing 10 fold cross-validation on our dataset. Since the two classes (obstacle/non-obstacle) are so unbalanced, we also present 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 0.12 can be achieved even by ignoring all the features.

In this experiment the color classifier performed extremely well, followed by the laser features and the texture which was mostly irrelevant due to the same problem as in the previous experiment. 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 indicates that the problem lies in the exponential penalty that AdaBoost "charges" for small classification margins. Our dataset has noise both in the features and in the labels, and these conditions are known to make AdaBoost overfits the data. A solution to this problem would be to use "soft-margin" AdaBoost variations such as the one described in [21].

2) Negative obstacle detection: Figure 8 shows results on the negative obstacle detection task. The obstacle is a large (0.75 m x 3 m x 1 m) rectangular depression in the ground, located at 3 m from the vehicle. Data collected by the color camera and the laser is used by three classifiers (color, texture, and 3-D laser statistics). We compared our best classifier fusion algorithm (stacked generalization) to a neural network having access to the entire feature vector. The results are consistent with our two previous experiments: stacked generalization and the network using all features reach comparable results, which are better than those of the best input classifier. Again, the two methods learn to ignore the texture-based obstacle detector.

IV. 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 feasible.

Our current efforts focus on developing specialized classifiers and on performing classification experiments in several other environments. We are also experimenting with more complex classifier combination schemes such as hierarchical mixtures of experts [22].

V. 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*.

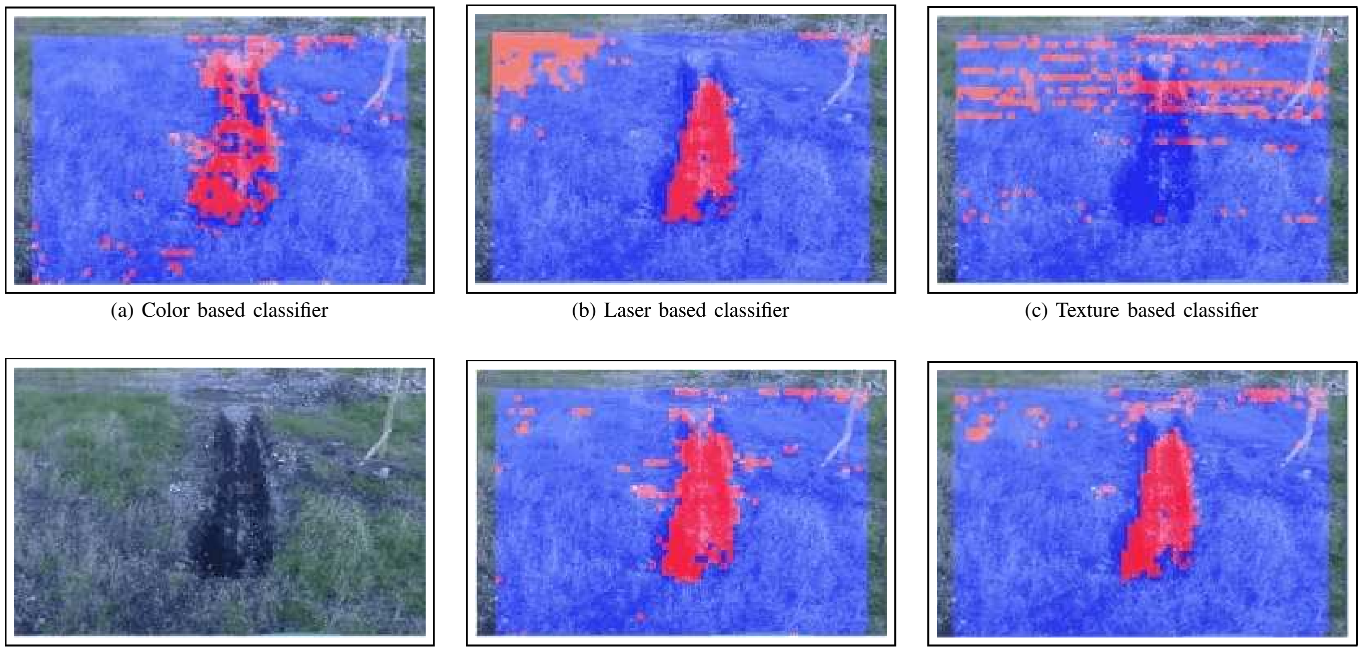
[2] C. Shoemaker and J.Bornstein, "The Demo III UGV program: A testbed for autonomous navigation research," in *Proceedings of the 1998 IEEE ISIC/CIRA/ISAS Joint Conference*.

[3] M. Ollis and T. M. Jochem, "Structural method for obstacle detection and terrain classification," in *Unmanned Ground Vehicle Technology*, 2003.

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

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

[6] 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.


(a) Color based classifier (b) Laser based classifier (c) Texture based classifier

(d) Scene (e) Neural network using all features (f) Stacked generalization

Fig. 8. Example of obstacle detection with the CMU vehicle. The blue and red areas correspond respectively to common sensor field of view and obstacles.

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

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

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

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

[11] 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.

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

[13] 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.

[14] 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.

[15] 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.

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

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

[18] T.-H. Hong, T. Chang, C. Rasmussen, and M. Shneier, "Feature detection and tracking for mobile robots using a combination of ladar and color images," in *Proceedings of the 2002 IEEE International Conference of Robotics and Automation*, Washington, D.C., May 2002, pp. 4340–4345.

[19] 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.

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

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

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