Proceedings of the 2007 IEEE/RSJ International
Conference on Intelligent Robots and Systems
San Diego, CA, USA, Oct 29 - Nov 2, 2007

Long-Term Learning Using Multiple Models For Outdoor Autonomous Robot Navigation

Michael J. Procopio, Jane Mulligan, and Greg Grudic
Department of Computer Science, University of Colorado at Boulder
{*procopio, janem, grudic*}*@cs.colorado.edu*

***Abstract*—Autonomous robot navigation in unstructured outdoor environments is a challenging area of active research. The navigation task requires identifying safe, traversable paths which allow the robot to progress toward a goal while avoiding obstacles. One approach is to apply Machine Learning techniques that accomplish *near to far learning* by augmenting near-field Stereo to identify safe terrain and obstacles in the far field. Some mechanism for applying *past learned experience* to the active navigation task is crucial for effective far-field classification. We introduce a new method for long-term learning in the robot navigation task by selecting a subset of previously learned linear binary classifiers. We then combine their output to produce a final classification for a new image. Techniques for efficient selection of models, as well as the combination of their output, are addressed. We evaluate the performance of our technique on three fully labeled datasets, and show that our technique outperforms several baseline techniques that do not leverage past experience.**

I. INTRODUCTION

Autonomous robot navigation in unstructured outdoor environments is a challenging area of active research. The navigation task requires identifying safe, traversable paths which allow the robot to progress toward a goal while avoiding obstacles. Stereo is an effective tool in the near field, but for smooth long-range trajectory planning or fast driving we need an approach to understand far field terrain as well. One approach to the problem is to apply Machine Learning techniques that accomplish *near to far learning* by augmenting near-field stereo readings with learned classifications of the appearance of safe terrain and obstacles in the far field.

Approaches which use image appearance or color to segment regions of interest for navigation have existed since the 1980s [1], [2], [3], [4], [5]. More recently programs such as DARPA's Learning Applied to Ground Robots (LAGR) have inspired work on using Machine Learning approaches to exploit image color and texture for classification of "traversable terrain" and obstacles in the far field [6], [7], [8], [9]. Although a number of the DARPA LAGR teams use Machine Learning techniques with color-based image features to build terrain models [10], [11], [12], none of them learn over time with multiple models, central to the proposed technique. The use of model ensembles has also been previously described in the literature [13], [14], [15], though not in the context of this particular problem.

The first author gratefully acknowledges the support of Sandia National Laboratories in sponsoring his dissertation research. Sandia National Laboratories is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the U.S. Department of Energy under contract DE-ACO4-94AL85000.

This work was partially supported by NSF CNS-0430593.

Robot hardware and support provided by the DARPA LAGR Program (DOD AFRL award no. FA8650-07-C-7702).

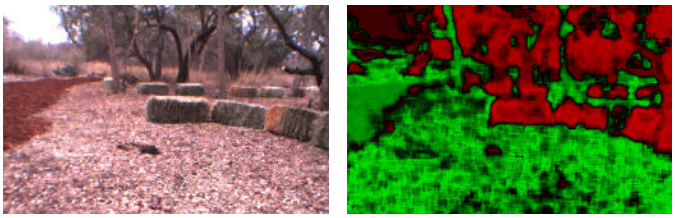

Fig. 1. Typical outdoor navigation scenario (left). Terrain classification on same image using proposed technique with 20 models (right).

Our previous work [16] frames this problem as a supervised Machine Learning problem. For each image acquired from the robot's vision system, stereo information is used to label regions of the image in the near field as traversable or non-traversable. That labeling information is then used to train a linear binary classifier which in turn is used to evaluate the remaining unlabeled pixels in the image. One problem with this technique is that with a single-model-per-image approach, there is no way to identify obstacles in the far field unless there are examples of those obstacles in the near field. For example, with this baseline approach, in order to plan around a patch of dense shrubbery seen in the far field, there must be examples of that foliage visible in the near field.

In this paper we describe a framework for long-term learning using multiple models built over time. Instead of building one model per image and then discarding it, our new approach is to build models *only as needed*, and *store them* in memory (persistent storage). The framework includes a technique for identifying models in memory that are appropriate for new images, and for combining the predictions of multiple models to arrive at a final terrain classification.

II. PROPOSED APPROACH AND MOTIVATION

This paper focuses on the problem of reliably classifying regions in the *far field*—outside of stereo range—as traversable, non-traversable, or varying degrees in between. The core principle of the proposed approach is to use *memory*. We must leverage past experience from previous navigational tasks, either from the current *mission* (A-to-B driving task), or from a prior mission perhaps many months ago. If a robot learns to discriminate between a tree and surrounding flat terrain by coming within stereo range of that scenario, then it should store this model and apply the knowledge to future tasks. The goal is to apply prior learned knowledge to classify terrain and identify obstacles in the far field *for which we might not have examples in the present near field to learn from.*

We identify four key challenges in storing and utilizing multiple models:

Model Structure: What information does a model consist of, and how can we store those models?

Model Selection: Given a new image classification task, how do we select a tractable subset of models that are appropriate to the image and will perform correctly?

Model Combination: Given a relevant set of models, how do we combine their predictions to arrive at a single classification?

Computational Efficiency: How do we select, evaluate, and combine the outputs of multiple models in real time?

Each of these challenges is addressed as we describe our framework for robot navigation using multiple models.

A. Model Structure

Our approach is to create large numbers of highly specialized *linear models*, augmented with histogram models to determine model relevance at a given test point. In their canonical form, a linear model in d dimensions consists of a vector of d coefficients, plus an offset [17]:

$$\beta = \langle \beta_1, \beta_2, \ldots, \beta_d \rangle^T, \beta_0$$

The model coefficients form a separating hyperplane that can be used to classify a d-dimensional test point $\mathbf{x} = \langle x_1, x_2, \ldots, x_d \rangle$. The output class $\widehat{y}$ predicted by the linear model at $\mathbf{x}$ is given by:

$$\widehat{y} = \operatorname{sgn}\left[\beta_0 + \beta \cdot \mathbf{x}\right] = \operatorname{sgn}\left[\beta_0 + \sum_{i=1}^{d} \beta_i x_i\right],$$

where $\widehat{y} \in \{-1, +1\}$.

Binary linear models for classification have two shortcomings for our scenario:

Binary vs. continuous/probabilistic output: Generally, linear binary classifiers return the signed distance from a test point to the separating hyperplane. This is usually binned and returned as one of two discrete class labels (e.g., -1 or $+1$). Planning systems can usefully exploit continuous traversability labelings interpreted as representing *degrees* of either safeness (traversable terrain) or lethality (obstacles). For example, not all obstacles (say, tall thick grass) are as "lethal" as other types (wide trees).

No concept of applicability: A standard classifier will produce an output for any input; however, it will not give an indication as to the *validity* of its output. For example, a classifier trained on images of rocks and grass will produce a dubious result for an image of a tree trunk. Given a problem that is known to be linearly separable, a linear model's output will be valid (trustworthy) *if that model was trained on data similar to the test point*. We refer to this concept as *model applicability*.

While techniques do exist for obtaining probabilistic outputs from some binary classifiers such as Support Vector Machines (SVMs) [18], these methods do not provide any notion of model applicability for a given test point.

Our technique addresses both of the above disadvantages by providing an indication of the relevance of model $\mathcal{M}$ for a test point $\mathbf{x}$ using *one-dimensional density estimation*. This model relevance value is used to provide both an indication of model applicability as well as a confidence in its prediction. The techniques for building and evaluating such a model are detailed below. Where applicable, experimental values for parameters used in this study are given in Table I.

B. Building a Model

1) To build a terrain classifier for an image I, we first extract a feature image I_f, which creates a d-dimensional representation of the image. Identifying the optimal features for terrain classification is an area of active research. Here, we use *color histograms* [19] which bin the color intensities in each of the three color channels (R, G, and B) in the neighborhood of the reference pixel. The number of bins b (here, fixed at 5) and the "window" dimension $c_w \times c_h$ (fixed at 7×7) are parameters of the color histogram feature extraction technique. Using 3 color channels and 5 bins per channel results in a feature image with feature depth d of 15 values (3 channels $\times$ 5 bins per channel).
2) Next, a random sample of training data consisting of n near-field image pixels from each class (as identified by Stereo) is selected ($Sample_1$). The resulting d-dimensional feature vectors (X_{lrn_1}) and their associated class labels ($\mathbf{y}_{lrn_1}$) are then used to train a binary linear classifier. We use a Linear Support Vector Machine (SVM), based on findings in [16]. From this linear classifier, the coefficients of the separating hyperplane (β) are extracted and stored.

$$X_{lrn_1} = I_f(Sample_1) = \langle \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N \rangle^T$$
$$\mathbf{y}_{lrn_1} = \textsc{Class-Of}(I(Sample_1)) = \langle y_1, y_2, \ldots, y_N \rangle^T$$
$$\beta = \textsc{Build-Linear-Model}(X_{lrn_1}, \mathbf{y}_{lrn_1})$$

3) Two more random samples of training data, disjoint from the data used to create the linear model, are extracted from the feature image ($Sample_2$ and $Sample_3$). $Sample_2$ and $Sample_3$ are homogeneous random samples of h_{gnd} traversable and h_{obs} non-traversable pixels, respectively. (Again, the class labelings come from Stereo.)

$$X_{lrn\_gnd} = I_f(Sample_2)$$
$$X_{lrn\_obs} = I_f(Sample_3)$$

4) For each of the two class samples, the data is evaluated by the linear classifier β created above, producing a vector $\mathbf{z}$ of model outputs (signed continuous values representing class and distance to the hyperplane). One vector will have mostly positive values, while the other will have mostly negative values reflecting the fact that the samples were chosen from different class distributions.

$$\mathbf{z}_{gnd} = (X_{lrn\_gnd} \times \beta) + \beta_0$$
$$\mathbf{z}_{obs} = (X_{lrn\_obs} \times \beta) + \beta_0$$

where $\mathbf{z} = \langle z_1, z_2, \ldots, z_N \rangle^T$.

5) $\mathbf{z}_{gnd}$ and $\mathbf{z}_{obs}$ are each used to build a histogram model ($\mathcal{H}_{gnd}$ and $\mathcal{H}_{obs}$). Each model bins the distances to the hyperplane; this can be thought of as the one-dimensional estimate of the density of hyperplane distance for a single class. During evaluation, this density estimate will provide a quantitative measurement for how "close" a test point $\mathbf{x}$ is to training data associated with that particular SVM/histogram model $\mathcal{M}$.

$$\mathcal{H}_{gnd} = \text{BUILD-HISTOGRAM-MODEL}(\mathbf{z}_{gnd})$$
$$\mathcal{H}_{obs} = \text{BUILD-HISTOGRAM-MODEL}(\mathbf{z}_{obs})$$

The final model $\mathcal{M}$ created for an input image I is therefore composed of a linear model (β), a histogram model for groundplane data ($\mathcal{H}_{gnd}$), and a histogram model for obstacle data ($\mathcal{H}_{obs}$):

$$\mathcal{M}(I) = \langle \beta, \mathcal{H}_{gnd}, \mathcal{H}_{obs} \rangle$$

Note that a model $\mathcal{M}$ can be constructed for every image I.

C. Evaluating a Test Point

For each new image the system must evaluate our appearance based classifiers for possibly thousands of unlabeled pixels. Classification in our framework involves selecting models from memory, evaluating those models, and combining each model's output to produce a "final" classification. The procedure is outlined below.

1) For each input image compute the color histogram feature image I_f. I_f has N d-dimensional "pixels" which form the test data matrix X_{tst}.

$$X_{tst} = I_f = \text{COLOR-HIST}(I_{RGB})$$

2) K models are selected to form the subset $\mathcal{S}$ (see Section II-D), including SVM models β and two histogram models $\mathcal{H}_{gnd}$ and $\mathcal{H}_{obs}$. Hyperplane coefficients for the K models form the $K \times d$ matrix β_{all} and $K \times 1$ vector β_0.
3) The entire $N \times d$ feature image data matrix (X_{tst}) is then evaluated through all K linear models to produce an $N \times K$ matrix of model outputs, Z_{all}.

$$Z_{all} = (X_{tst} \times \beta_{all}) + \beta_0 = \langle \mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_K \rangle, \mathbf{z} \in \mathbb{R}^N$$

4) The outputs from each linear model ($\mathbf{z}_i$) are then passed through each pair of histogram models and evaluated. This results in two $N \times K$ matrices of values on $[0, 1]$, which represent the confidence that test points belong to groundplane (traversable terrain) or obstacle (non-traversable terrain).

$$P_{all\_gnd} = \text{EVAL-HIST-MODELS}(\mathcal{H}_{all\_gnd}, Z_{all})$$
$$P_{all\_obs} = \text{EVAL-HIST-MODELS}(\mathcal{H}_{all\_obs}, Z_{all})$$
$$\text{where } P_{all\_*} = \langle \mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_K \rangle, \mathbf{p} \in \mathbb{R}^N$$

5) Finally, the two sets of outputs P_{all_gnd} and P_{all_obs} are combined to create a final classification (see Section II-E).

$$\mathbf{p}_{combined\_gnd} = \text{COMBINE-MODELS}(P_{all\_gnd})$$
$$\mathbf{p}_{combined\_obs} = \text{COMBINE-MODELS}(P_{all\_obs})$$
$$\text{where } \mathbf{p}_{combined\_*} = \langle p_1, p_2, \ldots, p_N \rangle^T$$

The resulting two vectors of combined probability estimates $\mathbf{p}_{combined\_gnd}$ and $\mathbf{p}_{combined\_obs}$ represent the final confidence for each pixel in I belonging to the respective class. This is the final output of the evaluation algorithm. The planning subsystem on the robot uses these two vectors to update the *cost map*.

Having *two* confidence values on $[0, 1]$—one for each class—allows us to subtract the values in $\mathbf{p}_{combined\_gnd}$ from those in $\mathbf{p}_{combined\_obs}$ yielding a new vector $\mathbf{q}$ of continuous values on the interval $[-1, 1]$, which can also be discretized:

$$\mathbf{q} = \mathbf{p}_{combined\_obs} - \mathbf{p}_{combined\_gnd}$$
$$\mathbf{q}_{binned} = \begin{cases} +1, & \text{where } \mathbf{q} \geq 0 \\ -1, & \text{where } \mathbf{q} < 0 \end{cases}$$

Thus we can represent the final synthesis of our evaluation of model $\mathcal{M}$ on a set of N test points X_{tst} by a single N-element vector $\mathbf{q}$ of values on $[-1, 1]$.

$$\mathcal{M}(X_{tst}) = \mathbf{q}$$

We can quantify the accuracy of this output directly by comparing it to $\mathbf{y}_{tst}$ which contains the class labels (-1 and $+1$) for X_{tst}. While standard binary accuracy can be calculated using $\mathbf{q}_{binned}$, using the continuous values in unbinned $\mathbf{q}$ suggests a new performance metric which we call *continuous classification accuracy*. Generally, the model receives a higher score at a test point $\mathbf{x}$ the more confident (i.e., closer to -1 or $+1$) it is on a correct prediction; similarly, it is penalized to a greater degree for a higher confidence incorrect prediction. This more general calculation is given by $|y_i + q_i|/2$ and has the property of returning the same result when evaluating $\mathbf{q}_{binned}$ against $\mathbf{y}_{tst}$ as does the standard binary accuracy calculation using $\mathbf{q}_{binned}$. We use this continuous accuracy when scoring models for selection (see Section II-D).

The subtraction operation used to produce $\mathbf{q}$ has some elegant properties. For example, if both histogram models predict an identical value, say, 0.8, then the final output q is 0, implying full uncertainty. This is appropriate, since the models are in conflict. The same applies to the case where neither model is confident and both histograms return a value of 0. The strongest result occurs when one histogram outputs 0, while the other outputs 1; this will result in value for q of -1 (full confidence, groundplane) or $+1$ (full confidence, obstacle).

A rendering of $\mathbf{q}$ appears in Fig. 1. Color (either green or red) represents terrain prediction (groundplane or obstacle), while intensity indicates classification confidence at that pixel. Accordingly, darker areas in the image represent areas where the model is uncertain, with black representing full uncertainty.

D. Model Selection

In general, we aim to maintain a large memory composed of hundreds (if not thousands) of models. Each model can be thought of as providing a specialized capability to discriminate between groundplane regions (safe, traversable areas) and obstacles (areas the robot should avoid). Not all models in memory will be relevant to the current image, and some may

even be detrimental to the overall classification of *I*. For example, some of the models in memory may have been trained on desert terrain, while the current mission traverses wooded terrain. Moreover, although there may be many models in memory appropriate for the image, there simply may not be time to evaluate them all. *Model selection* therefore becomes a critical question.

Generally, we want pick the best subset of models possible, where "best" means the subset $\mathcal{S}$ of *K* models that will result in the best navigational performance. Given some value for *K*, *how do we choose* $\mathcal{S}$?

We propose to choose $\mathcal{S}$ by first scoring each model on the image *I*, then ranking each model by score, and from this ranking, selecting the top *K* models. This implies that all *M* models in memory are evaluated in some manner with respect to *I*. In our approach, this evaluation occurs on the set of near-field stereo data $\mathcal{T}$ for which we have labels (and for which we can calculate model error). Since *M* can grow quite large, and $\mathcal{T}$ can contain tens of thousands of points, evaluation of each model $\mathcal{M}$ in memory on the entire set $\mathcal{T}$ is computationally prohibitive (see Section II-F). We therefore select a subset of *L* stereo data points called $\mathcal{T}_L$. *L* can scale up or down as computational requirements demand. Naturally there is a tradeoff between the value for *L* and the quality of the estimate of the score for $\mathcal{M}$ on *I*. Note that this estimate ($Score_{\mathcal{M}}(\mathcal{T}_L)$) comes from near-field labels only; an area for future work is to incorporate some prediction of far-field performance in this score.

Calculating the score for $\mathcal{M}$ on $\mathcal{T}_L$ is an ongoing area of research. Our current technique is to use the mean continuous classification accuracy (see Section II-C) of $\mathcal{M}$ on $\mathcal{T}_L$:

$$Score_{\mathcal{M}}(\mathcal{T}_L) = \frac{1}{L}\sum_{i=1}^{L}\frac{|y_i + q_i|}{2}$$

where $\mathcal{T}_L = \langle X_{tst}, \mathbf{y}_{tst}\rangle$, $q_i = \mathcal{M}(\mathbf{x}_i)$,
$X_{tst} = \langle \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_L\rangle^T$, and $\mathbf{y}_{tst} = \langle y_1, y_2, \ldots, y_L\rangle^T$.

E. Model Combination

Given a test point $\mathbf{x}$ and a subset $\mathcal{S}$ containing *K* models selected from memory, how do we arrive at a final classification for $\mathbf{x}$ that considers each model's output? Combining the outputs from multiple models is also a current area of research. We have devised an initial basic technique, described below.

As described above P_{all_gnd} and P_{all_obs} are constructed during model evaluation. Here, we consider only P_{all_gnd} for clarity. We wish to combine the K model outputs for each point into a single value, such that we end up with a final condensed *N*-element column vector $\mathbf{p}_{combined\_gnd}$.

The best mechanism for combination is still to be determined. Our current mechanism is to take the simple (unweighted) average over the values in each row in P_{all_gnd}, thus giving each model equal weight in the final output. As a special case, we do not include values of 0 in the mean, since 0 is a special value in the histogram evaluation that indicates the model was not trained on data similar to $\mathbf{x}$ and is therefore not applicable. The divisor in the mean operation (nominally *K*) is adjusted to reflect the number of pruned values of 0 and is called K_p.

$$\mathbf{p}_{combined} = \frac{1}{K_p}\sum_{i=1}^{K_p} p_i,\ p_i \neq 0$$

Possible alternative approaches involve combining model outputs by taking the *maximum* value over the *K* outputs (i.e., the most confident model dominates), or combining the outputs using weights from a log-odds function [20] such that models with extremely high or low confidence outputs will dominate. Another option could involve taking the weighted mean where the weights are the actual scores from each of the *K* models, or using weights derived from some notion of temporal relevance (e.g., the most recent models could be weighted more heavily than older models). A final idea would involve keeping a running score for each model, with weights based on each model's overall historical usefulness.

TABLE I
SUMMARY OF PARAMETERS AND THEIR VALUES IN THIS STUDY

Param	Definition	Study Value
M	Total number of models in memory	100
K	Number of models to be selected	(variable)
d	Dimension of feature vector	15
N	Number of test points in an image	320×240
L	Number of test points in stereo subset	100
n	Num. training ex./class for linear model	100
b	Num. bins/channel for color histogram	5
c_w, c_h	Width, height of color hist. window	7×7
h_{gnd}, h_{obs}	Num. histogram training examples	All remaining

F. Computational Efficiency

Robot navigation is done in real time, and must be achieved within the constraints of the robot's limited onboard computational ability. A general sketch of computational demands of our approach follows. (Values used in this study for the parameters referenced below are listed in Table I.)

Evaluating *N* *d*-dimensional test points through a linear model is $N{\times}d$ multiplications. Evaluating *K* such linear models is therefore $K{\times}N{\times}d$ multiplications. Evaluation of the *N*-element vector $\mathbf{z}$ (output by the linear model) through a single histogram model takes O(N) time. Training an SVM model is classically an O(n^3) procedure; faster methods are available for linear kernel SVMs [21] although LIBSVM (used in this study) makes no such optimizations.

Model selection introduces further computational demands. In the proposed technique, *all M* models in memory are evaluated on a small subset of stereo data $\mathcal{T}_L$ containing *L* test points; this requires $M{\times}L{\times}d$ multiplications for the linear model evaluation and an additional $2{\times}M{\times}L$ operations to evaluate all *L* points through both histograms for each model. Combining models using the proposed method typically involves *N* mean calculations over *K* values each, an O(NK) operation.

G. Summary of Parameters and Symbols

Our approach incorporates a number of tunable parameters whose values have implications for both computational demands and the ultimate navigational performance of the robot. A challenging area of research is to find a principled method for optimizing the values for the parameters listed in Table I such that the robot's navigational performance is maximized while still staying within some real-time frames/second performance requirement. Table I summarizes the tunable parameters in our approach and their fixed values used in the experiments for this paper. The values listed, when fixed, have been hand-chosen based on past experience with their performance characteristics.

III. EXPERIMENTAL DESIGN

A. Research Objectives

Our experiment design is driven by the following questions:

- Does the memory-oriented multiple model approach proposed here outperform baseline techniques, e.g., using SVM on a one-model-per-image basis?
- Is our method for selecting models effective?
- How does the performance of our multiple model technique vary with the number of models selected? How many models should we use?

Note that for this study, we do not experimentally vary the model combination method. Informal testing has shown that the model combination technique outlined previously (Section II-E) performs adequately; this is an area for future research.

B. Experimental Approach

To answer these questions, an experimental framework was developed. The following points are central to this framework:

Real data: Experiments are to be performed on image sequences taken from outdoor scenarios using standard hardware found on existing robot platforms.

Varied datasets: Different terrains pose different problems, and a variety of terrain, seen under different lighting conditions, is necessary to fully test any approach.

Hand-labeled "ground truth" images: To produce meaningful performance metrics and comparisons we require ground-truth data. In this study we evaluate the output of our technique against test images hand-labeled by a human, which means all parts of the image (not just the near field) are considered in the evaluation.

Comparison to baseline techniques: There are a number of simpler frameworks that can address model selection and combination; we compare our proposed technique to the performance of several of these baseline approaches.

Randomized experiments: In our approach randomness plays a role when selecting the training data for both the linear models and the two histogram models. There is a further random component when selecting the subset $\mathcal{T}_L$ of labeled data used for scoring during model selection. In this paper 16 randomized experiments are used to determine average and variability of performance. Accuracy values in this study have a typical standard deviation of 0.5–1.0% over all randomized experiments; this variation is small enough to where error bars are not shown graphically on the plots.

C. Baseline Techniques

The multiple model technique is compared against two baseline techniques.

One-model-per-image SVM: For every image *I*, supervised learning is used to create a linear model for that image, using training data labeled by stereo. This model is then used to classify the remainder of the image (including the far field), and is discarded afterward.

One-model-per-dataset SVM: Similar in principle to above, however, only one model is built using the *first* frame from the dataset. This model is used to classify both the first image and all remaining images in the dataset.

D. Datasets

For the experiments presented in this paper, we have extracted image frames from log files recorded during live runs of the robot in the DARPA LAGR competition. Each dataset consists of very different appearance and terrain, covering different lighting conditions (shaded, natural, and high intensity with shadows). The terrain varies greatly, with combinations of ground plane type (mulch vs. dirt vs. woods), foliage, natural obstacles (trees, dense shrubs) and man-made obstacles (hay bales).

Each dataset consists of 100-frame image sequences. Each image was manually labeled, with each pixel being placed into of three classes: Obstacle, Groundplane, or Unknown. If it was difficult for a human to tell what a certain area of an image was—even when using context—then that region was labeled as Unknown.

The datasets and their human labelings have been made publicly available in MATLAB format in order to encourage further experimentation. Raw stereo disparity information is also included to facilitate experiments with different near-field labeling techniques. The datasets can be found online at [22].

E. Performance Quantification

Performance quantification is reported as binary classification accuracy (1 – *error*). For a model (or combined model) $\mathcal{M}$ evaluated on a labeled test image *I* consisting of *N* test points (X_{tst}) and their associated class labelings ($\mathbf{y}_{tst}$), the accuracy $Acc_{\mathcal{M}}$ of $\mathcal{M}$ on *I* is the proportion of test points in *I* correctly predicted by the model:

$$Acc_{\mathcal{M}}(X_{tst}, \mathbf{y}_{tst}) = \frac{1}{N} \sum_{i=1}^{N} \begin{cases} 1, & \text{if } \mathcal{M}(\mathbf{x}_i) = y_i; \\ 0, & \text{otherwise.} \end{cases}$$

where $X_{tst} = \langle \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N \rangle^T$, and $\mathbf{y}_{tst} = \langle y_1, y_2, \ldots, y_N \rangle^T$. This calculation adapts to both the multiple model approach (using $\mathbf{q}_{binned}$, see Section II-C) and the baseline approaches by binning the linear model outputs $\widehat{y}$ (Section II-A).

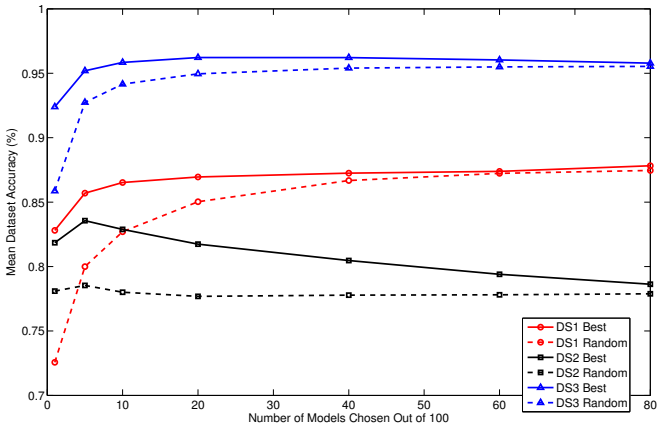

Fig. 2. Comparison of Model Selection Method *Best-K* vs. *Random-K*

F. Experimental Procedure

For each dataset, one model is built for each of the 100 images using the near-field stereo information associated with that image; this results in a model store of 100 distinct models for the experiment. The single-model-per-image baseline technique proceeds by classifying image I_t with the linear model β_t previously built from that image. The one-model-per-dataset baseline technique takes the first linear model β_1 (trained from the first image in the dataset, I_1), and uses this single model on the first image and all subsequent images in the dataset.

Our multiple model technique uses the entire set of 100 prebuilt models for all images. If it chooses $\mathcal{M}_t$, then the linear model β_t contained therein is the same linear model used by the single-model-per-image technique at I_t. Although having "future" information (i.e., a trained model) from image I_{t+1} available to help classify image I_t may seem unrealistic, in practice this is exactly the scenario we design for (e.g., the robot has previously navigated the path). The experiments are designed to measure the multiple model technique's ability to leverage this past information by selecting and combining models effectively.

All experiments were conducted in a MATLAB R2006b environment; LIBSVM v2.83 for MATLAB [23] was used for the linear SVM implementation.

IV. EXPERIMENTAL RESULTS

A. Effectiveness of Model Selection Technique

We compare our model selection technique *Best-K* (described in Section II-D) to a random selection (*Random-K*). For this experiment, trials were run for $K = \{1, 5, 10, 20, 40, 60, 80\}$.

The results are summarized graphically in Fig. 2. For all three datasets, the best performance was achieved when using the proposed scoring technique for selection (*Best-K*) compared to the baseline random subset selection of the same number of models (*Random-K*). Depending on the dataset, for K=10 models, classification accuracy was 3–5% higher when scoring and selecting models compared to a random selection. This trend holds true over all datasets. Figure 2 also illustrates that as the number of models increases, the two scores converge. Since our model store (memory) is fixed at 100 models (one model per image), as K increases, the set of models selected for the *Best-K* method intersects more and more with the set selected for the Random-K method. As this overlap increases, we naturally expect the performance results to converge.

Another result demonstrated in Fig. 2 is the effectiveness of combining models. A significant performance gain is achieved by combining 10 or 20 models versus combining 5, and especially compared to using 1 model (even the "best" one). Most interesting is that there are diminishing returns with respect to performance as the number of models increases. In fact, combining somewhere between 10 and 20 models produces close to maximal performance on these datasets. Our explanation is that 10 or 20 specialized models are sufficient to cover the problem "space"; it is likely that many models overlap (i.e., have very similar separating hyperplanes). This is not surprising given that many images in a sequence are similar.

These data suggest that overall performance may be hindered by forcing a selection of *too many* models. Experimentally this behavior is observed in Dataset 2; performance declined roughly linearly by about 5% as the number of models selected increased past 5 (here, the optimal value) through 80. This result suggests that forcing a selection of the K best models may not be optimal. For example, if K were fixed at 20, but only 5 models in memory were really applicable, then performance might suffer as a result. An immediate idea would be to pick models whose score falls above some threshold T, however some frames are simply "harder" than others for appearance-based classification and T may need to vary accordingly. Knowing when to build a new model vs. when to use existing models in memory is an important area of future research.

B. Effectiveness of Multiple Model Approach

Our experiments show that for two of the three datasets (Datasets 1 and 3), the multiple model approach performs significantly better than the baseline (3–5% improvement in classification accuracy). For the other dataset (Dataset 2), it performed slightly worse (3%). These results are summarized graphically in Figures 3 through 6. Overall numerical results are provided in Table II.

Both qualitatively and quantitatively, Dataset 2 has some of the most challenging terrain for appearance-based classification due to the bright, intense lighting conditions and the heavy presence of shadows; this dataset is particularly challenging since shadows on the ground plane have appearance similar to the dominant obstacle (dark shrubbery). This dataset also has the property that there were near-field stereo examples of most of the obstacles present in the far field, an ideal scenario for the one-model-per-image approach. We credit this for the baseline technique's improved performance over the multiple model technique (for Dataset 2).

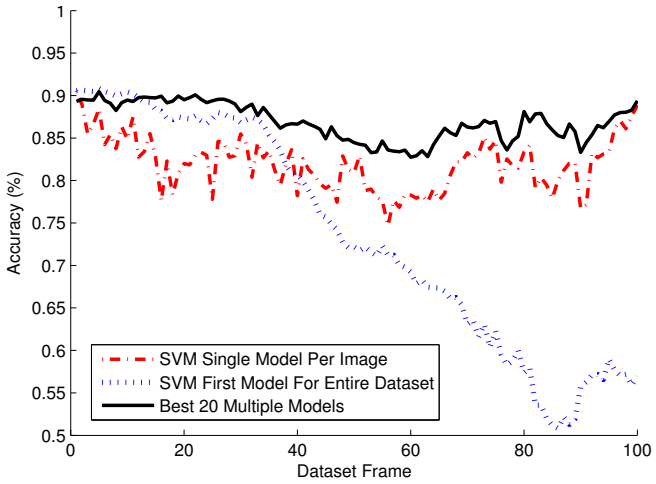

Fig. 3. Baselines vs. Multiple Model Method, Dataset 1

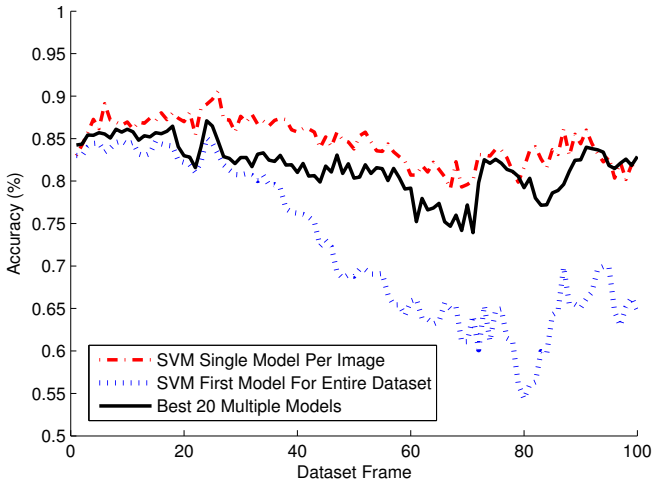

Fig. 4. Baselines vs. Multiple Model Method, Dataset 2

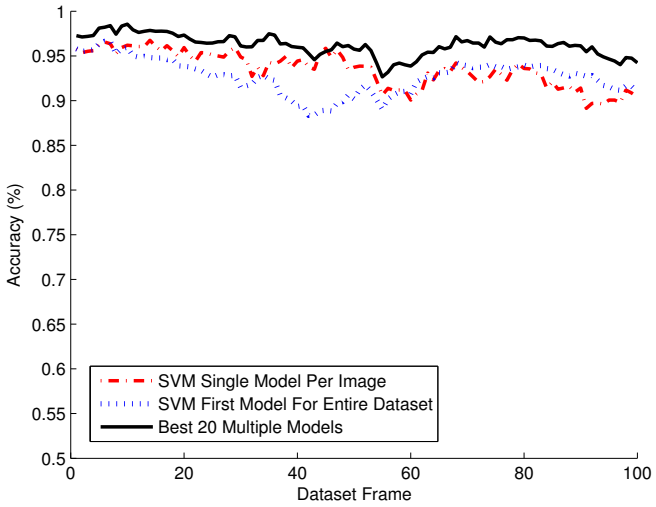

Fig. 5. Baselines vs. Multiple Model Method, Dataset 3

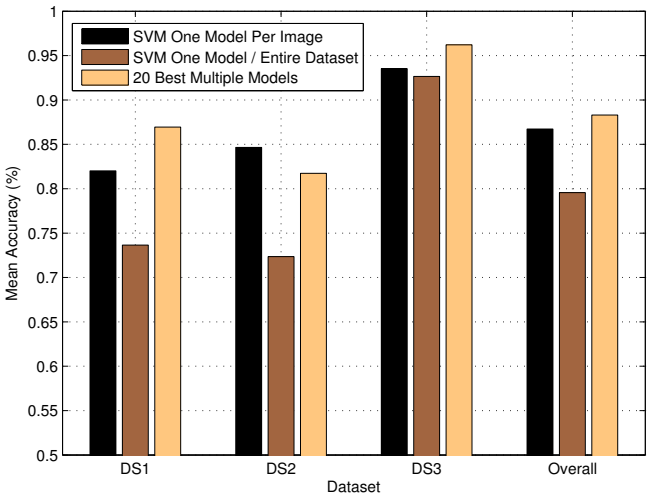

Fig. 6. Baselines vs. Multiple Model Method, Overall Performance

TABLE II
OVERALL RESULTS - MEAN CLASSIFICATION ACCURACY (%)

Method	DS1	DS2	DS3	Overall
SVM - One Model Per Image	82.02	84.65	93.53	86.73
SVM - One Model Per Dataset	73.66	72.36	92.65	79.56
Best 20 Multiple Models	86.95	81.74	96.22	88.30
Mean	80.88	80.02	94.13	

A related experiment involved creating a single linear SVM model for the first image, and then using this one model for every remaining image in the dataset. The results show that, as expected, this does not result in strong overall classification performance, except on frames with similar appearance to the frame on which the model was originally built (Figs. 3, 4, and 5). This underscores the notion that each linear model is a very specialized model and should only be used when applicable.

Figure 6 and Table II summarize the overall performance on the three datasets involved in this study. These are mean accuracy values over all frames, weighted by the number of valid expert-labeled test points in each frame. Overall, our multiple model approach performed better than the baseline by about 1.6%, a 12% reduction in error. As expected, the "one model per dataset" baseline method performed the worst overall. However, that technique did perform quite well on the early (30+) frames of each dataset, "near" to the image on which the single model was trained. This is an important result in that one model may be perfectly adequate for as long as the terrain stays similar to the terrain on which the model was trained.

V. CONCLUSIONS AND FUTURE WORK

This paper addresses the open problem of image based navigation in unstructured environments. Stereo data in the near field are used to learn models that use image appearance (color or features) to classify obstacles and traversable terrain in the far field. We propose saving and reusing learned models over time to make the process more efficient. This model memory is motivated by the need to classify far-field terrain

without the need for examples of the same terrain appearing in the near field of the same frame.

Using multiple models for classification involves challenges in model structure, model selection, model combination, and computational efficiency. We describe methods that address all four of these challenges, forming a framework that can leverage past experience (memory) stored as discrete, highly specialized linear binary classifiers. We perform a rigorous evaluation of this technique, comparing it to a baseline method that involves using just one linear binary classifier learned for each image. Based on results for hand-labeled ground truth datasets, we conclude that overall using multiple models results in higher classification accuracy. We also provided experimental evidence that the proposed technique for model selection is viable, outperforming a random selection of the same number of models. Finally, we gain interesting insight from experiments where the number of models selected is varied, demonstrating that only a relatively small number of models are needed to achieve robust classification results.

The main contribution of this paper has been to demonstrate that multiple models learned over time are an effective mechanism for classifying far-field terrain.

Future work will focus on improved techniques for selecting models, in particular ways to select models with expected high performance in the far field. Model selection methods such as "Previous-K" which combine the models from the K most recent frames should be evaluated. More intricate and theoretically motivated methods for combining models will also be explored (e.g., Bayesian Model Averaging [24]). Experiments will be conducted using these and other general ensemble methods. Additional baseline approaches will be considered, e.g., voting via multiple SVM models. Future efforts will develop well-founded techniques to determine when to build a new model vs. when an adequate subset of models exists in memory.

In future experiments, new datasets will be used. These will consist of frames from the same three scenarios presented in this paper, but each scenario will contain four distinct sequences of images, two each from two different lighting conditions. Further, instead of only 100 labeled frames, each sequence will consist of 300–500 labeled frames, allowing for concept drift to manifest more clearly. Finally, each frame in the new datasets will be larger, with 640×480 resolution.

All datasets used in experiments in this paper are available on the web at [22]; the additional datasets and hand-labelings noted above will also be made available in the future.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the contribution of Sandia National Laboratories, the National Science Foundation, the DARPA LAGR program, Daniel Lee, Sam Reid, and reviewers' comments.

REFERENCES

[1] R. Wallace, K. Matsuzaki, J. Crisman, Y. Goto, J. Webb, and T. Kanade, "Progress in robot road-following," in *Proc. of 1986 IEEE International Conference on Robotics and Automation*, vol. 3, 1986, pp. 1615–1621.

[2] J. Crisman and C. Thorpe, "Unscarf, a color vision system for the detection of unstructured roads," in *Proc. 1991 Int. Conf. on Robotics and Automation*, Sacramento, CA, April 1991, pp. 2496–2501.

[3] D. Kuan, G. Phipps, and A.-C. Hsueh, "Autonomous robotic vehicle road following," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, no. 5, pp. 648 – 658, 1988.

[4] M. Turk, D. Morgenthaler, K. Gremban, and M. Marra, "Vits-a vision system for autonomous land vehicle navigation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, no. 3, May 1988.

[5] G. N. DeSouza and A. C. Kak, "Vision for mobile robot navigation: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 2, pp. 237–267, 2002.

[6] M. Happold, M. Ollis, and N. Johnson, "Enhancing supervised terrain classification with predictive unsupervised learning," in *Proceedings of Robotics: Science and Systems*, Cambridge, USA, June 2006.

[7] G. Grudic and J. Mulligan, "Outdoor path labeling using polynomial mahalanobis distance," in *Proceedings of Robotics: Science and Systems*, Cambridge, USA, June 2006.

[8] H. Dahlkamp, A. Kaehler, D. Stavens, S. Thrun, and G. Bradski, "Self-supervised monocular road detection in desert terrain," in *Proceedings of Robotics: Science and Systems*, Cambridge, USA, June 2006.

[9] R. Manduchi, A. Castano, A. Talukder, and L. Matthies, "Obstacle detection and terrain classification for autonomous off-road navigation," *Auton. Robots*, vol. 18, no. 1, pp. 81–102, 2005.

[10] L. Jackel, E. Krotkov, M. Perschbacher, J. Pippine, and C. Sullivan, "The DARPA LAGR program: Goals, challenges, methodology, and Phase I results," *Journal of Field Robotics*, vol. 23, pp. 945–973, November/December 2006.

[11] J. Sun, T. Mehta, D. Wooden, M. Powers, J. Regh, T. Balch, and M. Egerstedt, "Learning from examples in unstructured, outdoor environments," *Journal of Field Robotics*, vol. 23, pp. 1019–1036, November/December 2006.

[12] A. Howard, M. Turmon, A. Angelova, L. Matthies, and B. Tang, "Learned traversability for robot navigation: From underfoot to the far field," *Journal of Field Robotics*, vol. 23, pp. 1005–1017, November/December 2006.

[13] W. N. Street and Y. Kim, "A streaming ensemble algorithm (sea) for large-scale classification," in *KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, NY, USA: ACM Press, 2001, pp. 377–382.

[14] K. Nishida, K. Yamauchi, and T. Omori, "Ace: Adaptive classifiers-ensemble system for concept-drifting environments." in *Multiple Classifier Systems*, 2005, pp. 176–185.

[15] R. Caruana, A. Niculescu-Mizil, G. Crew, and A. Ksikes, "Ensemble selection from libraries of models," in *ICML '04: Proceedings of the twenty-first international conference on Machine learning*. New York, NY, USA: ACM Press, 2004, p. 18.

[16] M. J. Procopio, T. Strohmann, A. R. Bates, G. Grudic, and J. Mulligan, "Using binary classifiers to augment stereo vision for enhanced autonomous robot navigation," University of Colorado at Boulder, Boulder, CO, Tech. Rep. CU-CS-1027-07, April 2007.

[17] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. New York: Springer, 2001.

[18] J. Platt, "Probabilistic outputs for support vector machines and comparison to regularized likelihood methods," in *Advances in Large Margin Classifiers*, A. Smola, P. Bartlett, B. Schoelkopf, and D. Schuurmans, Eds., 2000, pp. 61–74.

[19] G. Stockman and L. G. Shapiro, *Computer Vision*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001.

[20] H. Moravec and A. E. Elfes, "High resolution maps from wide angle sonar," in *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, March 1985, pp. 116 – 121.

[21] T. Joachims, "Training linear SVMs in linear time," in *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, NY, USA: ACM Press, 2006, pp. 217–226.

[22] M. J. Procopio, "Hand-labeled DARPA LAGR datasets." [Online]. Available: http://ml.cs.colorado.edu/∼procopio/

[23] C. C. Chang and C. J. Lin, *LIBSVM: a library for support vector machines*, 2001, software available at http://www.csie.ntu.edu.tw/∼cjlin/libsvm/.

[24] J. Hoeting, D. Madigan, A. Raftery, and C. T. Volinsky, "Bayesian Model Averaging: A tutorial (with discussion)," *Statistical Science*, vol. 14, no. 4, pp. 382–417, 1999.