Improving Robot Navigation through Self-Supervised Online Learning

Boris Sofman, Ellie Lin, J. Andrew Bagnell, John Cole, Nicolas Vandapel, and Anthony Stentz
Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213
e-mail: bsofman@ri.cmu.edu, elliel@ri.cmu.edu, dbagnell@ri.cmu.edu, jgcole@ri.cmu.edu, vandapel@ri.cmu.edu, axs@ri.cmu.edu

Received 8 April 2006; accepted 1 November 2006

In mobile robotics, there are often features that, while potentially powerful for improving navigation, prove difficult to profit from as they generalize poorly to novel situations. Overhead imagery data, for instance, have the potential to greatly enhance autonomous robot navigation in complex outdoor environments. In practice, reliable and effective automated interpretation of imagery from diverse terrain, environmental conditions, and sensor varieties proves challenging. Similarly, fixed techniques that successfully interpret on-board sensor data across many environments begin to fail past short ranges as the density and accuracy necessary for such computation quickly degrade and features that are able to be computed from distant data are very domain specific. We introduce an online, probabilistic model to effectively learn to use these scope-limited features by leveraging other features that, while perhaps otherwise more limited, generalize reliably. We apply our approach to provide an efficient, self-supervised learning method that accurately predicts traversal costs over large areas from overhead data. We present results from field testing on-board a robot operating over large distances in various off-road environments. Additionally, we show how our algorithm can be used offline with overhead data to produce a priori traversal cost maps and detect misalignments between overhead data and estimated vehicle positions. This approach can significantly improve the versatility of many unmanned ground vehicles by allowing them to traverse highly varied terrains with increased performance.

1. INTRODUCTION

Autonomous robot navigation in unstructured natural environments has been demonstrated extensively in a large variety of terrain, sensor payload, and mission scenarios [see for example Kelly et al. (2006), Bodta and Camden (2004), and Goldberg, Maimone & Matthies (2002)]. Even though powerful at sensing, modeling, and interpreting the environment, these systems required significant tuning of parameters, either by hand or supervised training, to best adjust their algorithms to the local environment where the tests are conducted.

Contract grant sponsor: Defense Advanced Research Projects Agency (DARPA).
Contract grant number: MDA972-01-9-0005.

This highlights a common problem that arises in mobile robotics where potentially powerful sensor data and features are often difficult to take advantage of because they are situation or location specific. For instance, camera imagery can potentially detect unpaved road in the desert significantly farther than some ladar-based systems can. Detecting such roads from a distance proved to be a crucial component in Stanford Racing's winning Grand Challenge entry (Lieb, Lookingbill & Thrun, 2005b). Unfortunately, such data can prove very resistant to automated interpretation. In particular, classifiers that prove to be powerful indicators of road in a particular area often do not generalize to new conditions.

Similarly, outdoor robot navigation can benefit from the now widespread availability of high quality overhead imagery and elevation data from satellite and aircraft (Kelly et al., 2006; Silver, Sofman, Bagnell, Vandapel & Stentz, 2006). Presently, nearly the entire world has been surveyed at 1 m accuracy, with higher resolution (better than 25 cm accuracy) data available in certain areas. With this overhead data, many of the difficulties associated with autonomous robot operation can be alleviated, even with the coarsest of terrain resolution. Systems can then dispense with myopic exploration and instead pursue routes that are likely to be effective.

In much the same way, the complete use of sensor data (such as ladar), if interpreted correctly, can help a robot make better decisions and increase traversal speed through improved knowledge of the environment. Unfortunately, as the complexity of environments increases, a robot's perception system must be engineered to evaluate its environment by first computing a set of intermediate features such as ground slope, object density, and vegetation classification that, while accurate and generalizable, limit the effectiveness of such systems to a range of about 15 m due to the quick decline in coverage density (see Figure 1).

Unfortunately, leveraging these tremendous resources on an autonomous robot proves difficult. Building systems that can reliably interpret overhead image data or far-range sensor data is far from easy as even the smallest variations in lighting, season, terrain, or even sensor calibration can have significant effects on data. Additionally, such estimates must be well calibrated with other on-board perception estimates of the terrain, or system performance may suffer.

One way to address such limitation is through on-line self-supervised learning where the autonomous system adjusts itself via perception and interaction with the environment. In this article, we address the problem of learning and inferring between two heterogeneous data sources that vary in density, accuracy, and scope of influence. The objective is to generalize from one data source, viewed as a reliable estimate, to be able to work with another, which may be high performance (e.g., long range or high accuracy) but difficult to generalize to new environments. We frame the problem as a simple, linear probabilistic model for which inference results in a self-supervised online learning algorithm that fuses the estimates from the two data sources. We discuss the advantages of this framework including reversible learning, feature selection, data alignment capabilities, reliable use of multiple estimates, as well as confidence-rated predictions.

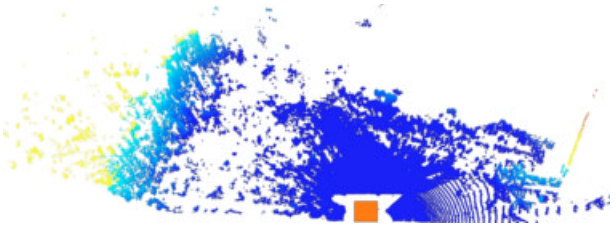


Figure 1. Typical ladar response from vehicle's perception system. Ladar points are color coded by elevation with lowest points appearing in blue and highest points appearing in yellow. Vehicle position is shown by the orange square. Notice the large drop in ladar response density (especially on the ground) as distance from the vehicle increases. Large objects such as the trees on the left generate ladar responses even at far ranges but are difficult to interpret through fixed techniques across different environments.

Furthermore, we demonstrate the approach in the context of long range navigation of a large unmanned ground vehicle during various field tests in complex natural environments. As it traverses an environment, the vehicle utilizes its on-board perception system and these difficult to interpret features (computed from overhead imagery and elevation data or far-range sensor data) to learn the mapping from these features to computed terrain traversal costs in order to predict traversal costs elsewhere in the environment where only overhead data or far-range sensor data are available, effectively extending the range of the vehicle's local perception system and allowing more effective navigation of the environ-

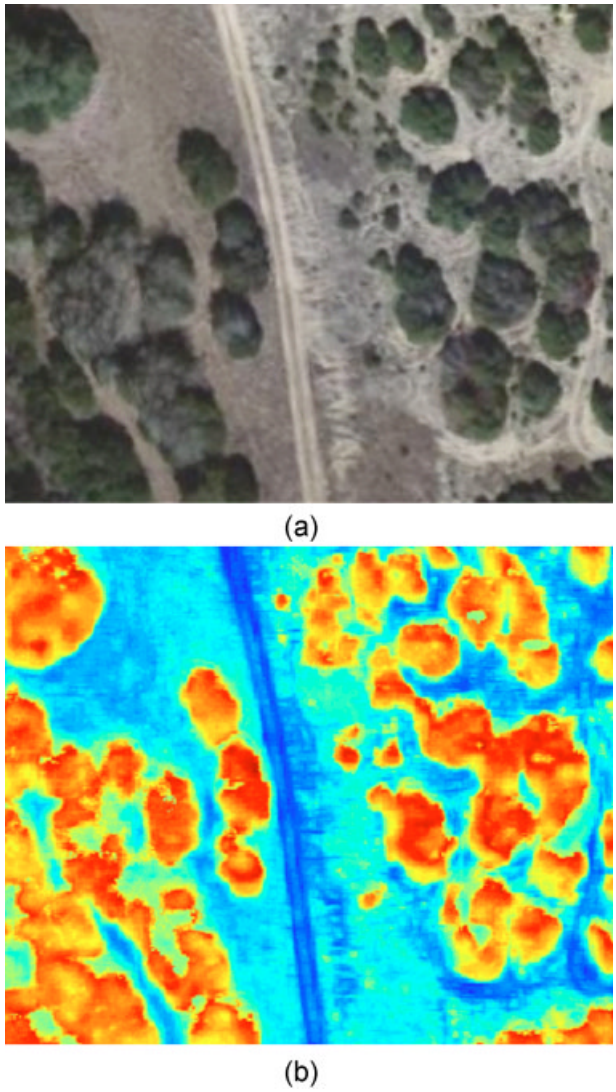


Figure 2. Sample results of terrain traversal cost predictions. (a) 0.35 m resolution color overhead imagery used by our online learning algorithm and (b) corresponding predictions of terrain traversal costs. Traversal costs are color-scaled for improved visibility. Blue and red correspond to lowest and highest traversal cost estimates, respectively.

ment (see Figure 2 for sample results).[1] This approach removes the necessity of human involvement and parameter engineering, which limits the versatility of many robotic systems.

Similar approaches have been demonstrated successfully in semi-structured natural environments (Lieb, Lookingbill & Thrun, 2005a). Numerous research efforts are also underway to take advantage of such techniques for obstacle avoidance using a monocular camera (LeCunn, Muller, Ben, Cosatto & Flepp, 2005), estimating the depth from monocular imagery (Michels, Saxena & Ng, 2006), terrain traversability classification (Kim, Sun, Oh, Rehg & Bobick, 2006), and slip prediction (Angelova, Matthies, Helmick, Sibby & Perona, 2006).

Section 2 presents the approach in detail. In Section 3, this approach is positioned in the context of ground robot navigation. Section 4 contains results from field tests in various environments with a large autonomous vehicle, as well as several offline applications of our algorithm.

2. APPROACH

2.1. Formalization

We approach the problem of leveraging the powerful, but difficult to generalize, features in a Bayesian probabilistic framework using the notion of scoped learning (Blei, Bagnell & McCallum, 2002). The scoped learning model admits the idea of two types of features: "global" and "local." Global features are generally useful, and their predictive power extends well to new domains, while local ones, which, although often very powerful, typically generalize poorly and are more difficult to take advantage of in a consistent way. These local features have *scope* that is limited to one particular domain. We wish to apply our system to extend the scope of such features to many possible domains. For our canonical problem of learning to leverage the extended range of overhead and far-range sensor data, these names may prove counter-intuitive, so we refer to them instead as *general* and *locale-specific* features. For the remainder of this article, "global" and "local" will refer to the proximity to the robot. Features generated from dense, vehicle-based ladar perception serve as our general features, while features generated from overhead-based imagery and elevation data and far-range sensor data serve as our locale-specific features. The latter are particularly valuable to mobile robots because of their extended range and widespread availability.

[1]This article is best viewed in color.

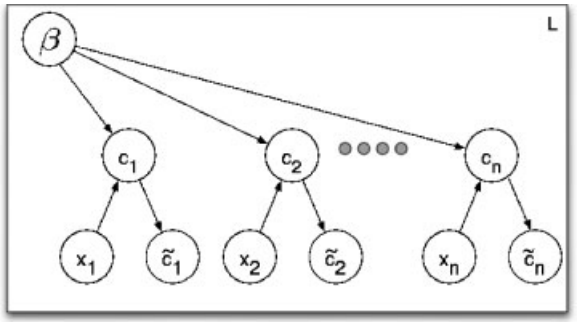


Figure 3. Graphical depiction of the scoped learning model. Hyper-parameters, including priors on the locale-specific parameters β and noise variances, lie outside the plate indexed by L and are not depicted.

2.1.1. Model

The scoped learning approach is a very simple probabilistic model (shown graphically in Figure 3) that captures this notion of features that have scope. The outer plate L represents in graphical model notation that there are independent locales in which the model will be applied (Jordan & Weiss, 2002). These correspond to new areas of the world in which our robot will operate.

Within the plate, we see a sequence of locale-specific features and corresponding general feature-based estimates. At each point in the sequence, we wish to make predictions about c (either all or a subset of them). Here, c is the true variable we wish to predict, and $\tilde{c}$ is an estimate of that variable coming from the general features, while x are our locale-specific features. The parameters β common to the locale (plate) capture the relationship between locale-specific features and the variables of interest c. The length of our sequence is n.

This learning model captures the idea of self-supervised learning (Kakade & Ng, 2005) in a Bayesian framework and extends the idea to integrate both the general feature-based estimates and the self-supervised locale-specific estimates. Driven by our application, we are particularly interested in the online *regression* case[2] where the goal is to infer the true continuous values c_i in an online fashion as general feature-based estimates $\tilde{c}_i$ become available.

[2]The original scoped learning work (Blei et al., 2002) was developed in the context of classification using discrete features, generative descriptions of those features, and in batch.

We choose a simple model for c as a function of the k locale-specific features $x=(x^1,\ldots,x^k)$ by modeling the distribution for c given x as a Gaussian with mean a linear function of x and with a variance of σ_l^2:

$$E(c|\beta,x) = \beta^T x. \tag{1}$$

We assume that the estimates from the general feature-based predictors have Gaussian noise and thus are distributed:

$$\tilde{c} \sim \text{Normal}(c,\sigma_g^2).$$

We take the σ_g and σ_l to be hyper-parameters lying outside the locale-specific plate.

2.1.2. Inference

We develop the inference for the model in an online fashion. Given a new data point $\tilde{c}_i$ estimating the true variable c_i, our goal is to compute new estimates[3] of the variables c_j, assuming we have already seen data $D=\{\{x\}_{1,\ldots,n},\{\tilde{c}\}_{1,\ldots,i-1}\}$. We can compute this by integrating over the uncertain parameters β, which describe the relationship between the true variable and the local features:

$$p(c_j|\tilde{c}_i,x_i,D) = \int d\beta p(c_j|\beta,\tilde{c}_i,x_i)p(\beta|\tilde{c}_i,x_i,D).$$

We can compute the required distribution over β as

$$p(\beta|\tilde{c}_i,x_i,D) \propto p(\beta|D)\int dc_i p(\tilde{c}_i|c_i)p(c_i|\beta,x_i).$$

In our linear-Gaussian model, this can be understood as revising the posterior distribution from $p(\beta|D)$ in light of a Gaussian likelihood that takes into account noise from both general and locale-specific features.

Our computation of the posterior distribution $p(\beta|\tilde{c}_i,x_i,D)$ is as follows. We first initialize our distribution to the prior distribution $p(\beta)$. Then, for every training example i, we multiply our distribution

[3]We assume that our prior on β is a priori independent of the features x so that inference will remain the same even in the case where the features become available in some sequence.

by $p(\tilde{c}_i|\beta,x_i)$. Since the prior distribution and $p(\tilde{c}_i|\beta,x_i)$ are normal, the posterior distribution is also normal. We use the notation $\hat{\beta}$ to represent the mean of the posterior distribution and V_β to represent the variance. Thus, computing $p(\beta|\tilde{c}_i,x_i,D)$ is performing a self-supervised learning using a Bayesian linear regression model with noise variance $\sigma_l^2 + \sigma_g^2$.

We use our current estimate of the posterior distribution when we want to predict a future outcome c_j. We are interested in predictions in two cases: first, when we have no general feature-based estimate $\tilde{c}_j$ for a particular c_j and, second, when such an estimate is available. In the first case, the predictive distribution $p(c)$ has mean $c_p = x^T\hat{\beta}$ and variance $\sigma_p^2 = \sigma_l^2 + \tilde{x}^T V_\beta \tilde{x}$ (Gelman, Carlin, Stern & Rubin, 2004). When we also have an estimate $\tilde{c}_j$, inference combines these two estimates:

$$p(c_j) = \text{Normal}\left(\sigma_p'^2\left(\frac{c_p}{\sigma_p^2} + \frac{\tilde{c}_j}{\sigma_g^2}\right), \sigma_p'^2\right)$$

where

$$\sigma_p'^2 = \frac{1}{1/\sigma_p^2 + 1/\sigma_g^2}.$$

We note that it is possible to compute the posterior distribution in batch, but we prefer to maintain an estimate of the posterior distribution as we receive general feature-based cost estimates so that we may immediately apply our algorithm to new data.

2.2. Advantages of the Bayesian Learning Approach

Using the online Bayesian scoped learning model provides a number of important benefits.

2.2.1. Confidence Rated Prediction

The variance estimate provided by our algorithm for the probability of each c can be used as a metric of confidence in the prediction. If a situation arises in which we must choose which one of several predicted outcomes to trust, we could simply use the one with the smallest variance.

Algorithm 1: Hyper-parameter re-estimation procedure

```
1: procedure
2:    Initialize all α_k and σ_l^2
3:    while α has not converged do
4:       Compute the mean β and covariance V_β of the
         distribution of our weights β
5:       For all K features, γ_k ← 1 − α_k V_βkk
6:       For all K features, α_k ← γ_k / β_i^2
7:    end while
8:    return α
9: end procedure
```

Once our α_k values have converged, we can remove any feature x_k with a corresponding optimal α_k value that tends toward ∞. Since an α_k value controls the inverse variance of each weight β_k, an α_k value that tends toward ∞ implies that the mean of the weight β_k value tends toward 0. Thus, we can remove that feature x_k since its weight β_k value of 0 would remove its contribution to predicting the output $\tilde{c}$.

2.2.2. Learning of the Hyper-Prior and Feature Selection

Our algorithm depends on a number of hyperparameter terms that may be chosen based on data from multiple locales. We discuss ways to choose the noise variance terms σ_l and σ_g in Section 2.1 and the prior distribution on parameters β in Section 3.1. The prior distribution $p(\beta)$ is an isotropic Gaussian that is independent for each weight, and each weight is dependent on a shared hyperparameter α that moderates the strength of our belief over the values our weights β might take. That one α value controls the inverse variance of each weight β. We can modify our prior $p(\beta|\alpha)$ to consist of K hyperparameters, with each α_k independently controlling the inverse variance of each weight, and define hyperpriors over all the α_k values. We can then use Tipping's hyperparameter reestimation algorithm to do feature selection (see Algorithm 1) (Tipping, 2003). In this way, we can both automate feature selection and bias our algorithm to prefer certain features for new locales (Ng & Jordan, 2001).

2.2.3. Reversible Learning

A problem that often arises in online learning is the handling of multiple estimates of a particular quantity. For instance, in our canonical example, our general feature-based estimates $\tilde{c}_i$ may improve as we get closer and denser laser readings of the terrain. It is not appropriate to treat these as independent

training examples: while they may differ in their variance, they are generally highly correlated. Neither is it useful to simply take the first estimate available: often this is a poor substitute for all the data. In our model, since we maintain an exact posterior distribution that lies inside the exponential family, we may effectively *remove* the effects of training on a data point by dividing out the likelihood term we had used to include it in the posterior (Gelman et al., 2004). In this way, we always have an estimate of the posterior distribution of β using the current best estimate $\tilde{c}_i$. Minka has developed an alternate use of this "removal trick" for approximate inference (Minka, 2001).

3. APPLICATION TO GROUND ROBOTICS

3.1. Context

A natural application of our algorithm is to the improvement of autonomous navigation capabilities for unmanned ground vehicles. Our algorithm lends itself well to addressing many of the issues that arise due to the diversity in the environments and data that robotic systems must encounter. In many robotic systems, variables most relevant to an unmanned ground vehicle, such as terrain traversal cost, are computed directly by the vehicle's on-board perception system through the processing of high density ladar data gathered by on-board sensors. Techniques to process such data are often generalizable to many environments so that the vehicle's perception system does not require much adjustment when dealing with new terrain, yet the limited range for this type of data source is a major shortcoming (see Figure 1). While data sources such as overhead imagery are widely available at high resolutions, developing fixed techniques to process such locale-specific data sources is difficult because even though they may be extremely useful in specific areas, they do not generalize well to new domains due to variations in terrain, lighting conditions, weather, and even time of data gathering.

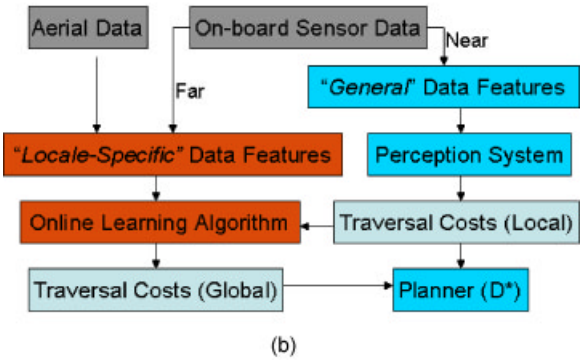
We demonstrate how our algorithm can be applied to the domain of mobile robotics in order to derive the most benefit from the availability of both general and locale-specific data sources without any human involvement. The performance of our algorithm is evaluated through actual field testing in a complex off-road environment on-board a rugged, all-terrain unmanned ground vehicle. Our results show how combining the adaptive performance of our algorithm with the inherent mobility of such a vehicle leads to more efficient navigation of complex environments. Additionally, we show several offline applications of our algorithm such as the ability to detect misalignments between overhead data and the vehicle's estimated position, a common problem in robotic systems that utilize such data. Finally, we demonstrate how our feature selection technique can be used to shorten the time required for training by reducing the feature space.

3.2. Terrain Traversal Cost Prediction

Our robot performs local sensing using ladar sensors and assigns traversal costs to the environment from features computed by interpreting the position, density, and point cloud distributions of sensed obstacles (these features generalize across most domains and therefore serve as our *general* features as defined in Section 2.1). The robot replans in real-time by finding minimum cost paths through the environment using the D* algorithm (see Figure 4) (Stentz, 1994). We demonstrate how our algorithm can learn to predict terrain traversal costs computed by the on-board perception system of our unmanned ground vehicle from both overhead data and far-range sensor data that are currently discarded. We also compare the predictive performance of our algorithm to that of a hand-trained classifier using superior data sources. We chose to predict traversal cost rather than intermediate results such as slope, density, or presence of vegetation because traversal cost is the metric that most closely governs a vehicle's navigation strategy through an environment. Our robot's perception system is proficient at effectively assessing terrain traversal costs, so it is desirable to be able to mimic its predictive abilities. We therefore use estimates from the robot's perception system to evaluate the accuracy of traversal cost predictions.

The characteristics of an environment change with varying conditions. However, even outdated overhead data can be useful since most distinct areas in an environment will maintain uniformity in their characteristics despite these variations. By relaxing restrictions on the recency of overhead data, our algorithm further increases its impact on improving robot navigation. Overhead data are relatively inex-

(a)



(b)

Figure 4. (a) Robot used for all field tests in its typical operating environment and (b) an illustration of how the online learning algorithm runs on-board the robot. Algorithm learns mapping from locale-specific data features (overhead or far-range sensor data) to locally computed terrain traversal costs (computed from general features) to make prediction elsewhere in the environment.

pensive and available at various resolutions for the entire world, so as the quality of global surveying improves, the applications of such research will greatly expand.

3.3. Features

3.3.1. Overhead Data

A set of feature maps for the vehicle's environment was generated from each overhead data source for use as input to the algorithm (these are our *locale-specific* features as defined in Section 2.1). In our implementation, HSV (hue, saturation, value) features were used to represent color imagery data while the pixel intensity of the black and white imagery data was used as a single feature. Raw RGB (red, green, blue) color data were inadequate for our approach due to their sensitivity to illumination variations.

A feature containing the maximum response to a set of ten Gabor filters at various orientations centered at each pixel was also generated to capture texture in each type of imagery. Additional features for the black and white imagery data were generated by computing the means and standard deviations in intensity within windows of 5 m around each pixel. Additional elevation-based features [similar to those described in Silver et al. (2006)] were computed when such data were present. All features were rescaled to the $[-1,1]$ range and a constant feature was also included.

Finally, clustering of all previously computed features was performed that allowed the algorithm to identify patterns in the feature input space that are relevant to the output regression. The Gaussian mixture model algorithm was chosen to cluster the input data because of its ability to generate membership features by assigning each data point a fractional degree of membership in each output cluster (see Figure 5) (Duda, Hart & Stork, 2000). Six clusters were chosen for our implementation.

3.3.2. Far-Range Sensor Data

Ladar and camera data were used to create the far-range sensor features used for training (once again, these are our *locale-specific* features as defined in Section 2.1). Ladar points in the environment were tagged with color values from the camera sensors by computing the pixel the ladar would appear in within the camera image. Color features were computed as described in the previous section. Additionally, the positions of the ladar points were used to compute the maximum vertical point spread and standard deviation of point heights. Additionally, in order to incorporate contextual information about the neighbors of a location, similar features were computed from the location's neighbors within a small window. A constant feature was included as well. As a robot travels toward a location, features for that location are computed regularly (to account for variations in features due to distance from the robot) and stored to be used as possible future training examples.

Finally, it should be noted that the *lack* of ladar

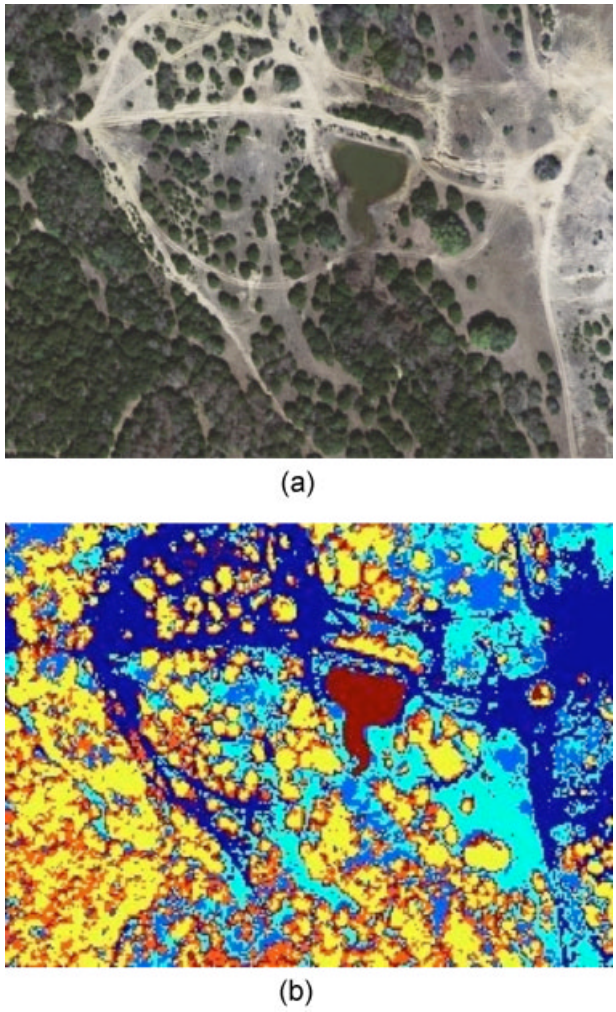


Figure 5. Sample clustering results from using the Gaussian mixture model algorithm on generated features. (a) Overhead color imagery data used to generate features and (b) resulting clustering into six clusters. Membership features were generated by computing the fractional degree of membership of each pixel in each cluster.

data in an area in front of the robot can serve as a confirmation of free space (since ladar hits are rarely available on road or grass past about 30 m due to the angle with respect to the sensor). We identify such free space by tracing away from the robot until a sensed object is encountered (up to 40 m away) and setting a tracing feature in all encountered cells. This feature is the sole feature for such areas.

Similar techniques may be used to generate features from any combination of data sources gathered through a variety of methods.

3.4. Training and Prediction

Traversal cost is difficult to quantify, so choosing appropriate values often requires careful engineering. In order to produce desired behavior when used with a path planning algorithm such as D*, traversal costs for undesirable areas such as heavy vegetation must be higher than traversal costs for ideal areas such as roads (our robot works with traversal costs in the range of 16 to 65535). For example, the robot's on-board perception system assigns traversal costs of 16 (the minimum) to roads while grass is assigned a traversal cost of 48, implying the robot would be willing to take a detour of three times the distance in order to stay on a road as opposed to driving over grass. Meanwhile, dense vegetation is often assigned traversal costs of over 10000 in order to encourage the robot to traverse elsewhere except under extreme necessity. Because of these traversal distance ratios, errors in traversal cost estimates in low-cost areas are more detrimental than similar errors in high-cost areas. An error of 100 to an area of extremely high traversal cost would have negligible effect, while the same error at an area of desirable terrain would radically change the behavior of the robot.

In order to work with a linear model, we deal with traversal costs within our algorithm on a logarithmic scale, converting from the normal traversal cost space for the purposes of training and prediction. The Gaussian error assumption embedded in our probabilistic model is a much better approximation when we measure error on this scale. Unlike in the regular traversal cost space, small errors in the log space lead to small errors in the traversal distance ratios.

Training examples are constructed from x_i, the vector of feature values (either from overhead data or far-range sensor data), and $\tilde{c}_i$, the average of all traversal cost estimates that have been calculated within the corresponding area. As with many robotic systems, the performance of our robot's on-board perception system quickly degrades as the distance from the robot increases (due to the lowered accuracy and density of sensor data), so the quality of a training example is measured by its proximity to the robot. Rather than struggling to decide at which point to utilize an example for training, the reversible learning capabilities of our algorithm allow us to maintain an optimal level of predictive abilities by ensuring that only the highest quality data available

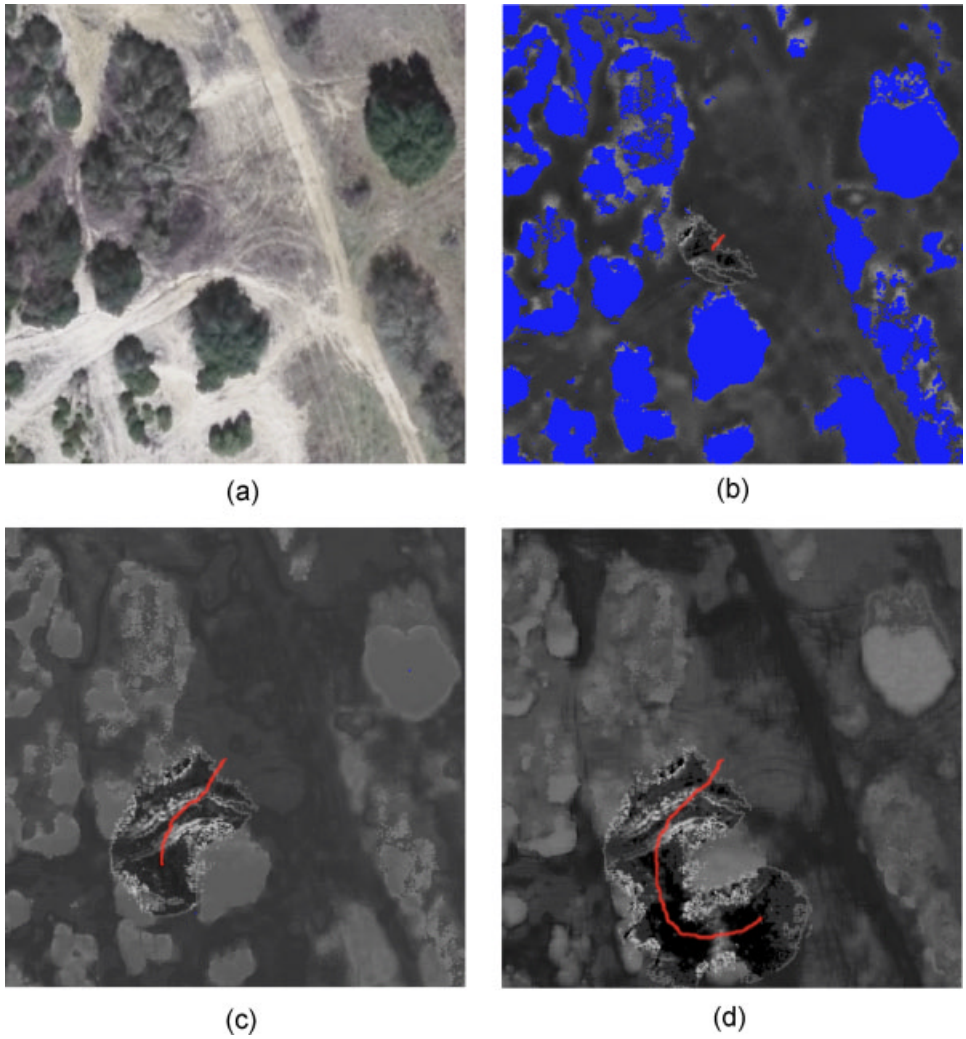


Figure 6. Training progress of online learning algorithm using overhead color imagery data for traversal of environment shown in (a) is shown in (b)–(d). Dimensions of shown areas are 150×150 m^2. Accumulated ground truth traversal costs computed by the robot's on-board perception system and vehicle path (shown in red) are overlaid on estimated traversal costs generated by the algorithm. Lower costs appear as darker colors and predictions that the algorithm lacked confidence in (due to insufficient representative training examples) are shown in blue.

impact its state. As the robot approaches locations that had previously been used for training, obsolete examples are *unlearned* in favor of higher quality training examples available for those areas. Estimates greater than 12 m from the robot are ignored since such estimates are very unreliable and would only corrupt the quality of training in cases where they cannot be replaced with better estimates. An example of this training process can be seen in Figure 6. As the vehicle explores more of the environment, the greater sample of training data allows it to more accurately interpret the locale-specific data. Notice how the shadow from the tree at the top right is initially estimated incorrectly as a very high-cost area [Figure 6(c)], but as the robot explores more of the environment, it begins to recognize its error and lower its estimate [Figure 6(d)].

As the algorithm acquires more training data, its predictive performance improves, allowing it to revise previously made traversal cost estimates. The

algorithm specifies a degree of confidence for each prediction based on the similarity of the example to past training data (as indicated by the variance estimate), so predictions in which the algorithm lacks confidence can be ignored in favor of an alternative source of predictions or a default value. Notice how in Figure 6(b) the algorithm is able to identify its estimates for the trees in the environment as areas of low confidence (shown in blue) until the robot first encounters the tree below its starting position.

3.5. Applications of Trained Algorithm

This algorithm can be used in a variety of ways to aid in unmanned ground vehicle navigation, both in real-time on-board a robot and offline once it has been trained. In the case of online terrain traversal cost prediction, the algorithm can be used to periodically update traversal cost estimates within a region around the robot where features have been computed so that a real-time path planning algorithm such as D* can revise the vehicle's global path to account for the changes. As shown in the following section, the use of this algorithm to extend the robot's field of view results in significantly shorter and more intelligent paths.

When using overhead data, one can make traversal cost predictions for a large area without ever having to traverse or acquire training examples from that area beforehand since the predictive state of the algorithm can be captured at any time by the vector β at that moment. Instead, as long as identical features are computed for the two areas, one can simply drive through a representative area for a short period of time in order to train the algorithm to make predictions in a much larger area. The following section will also show that a priori traversal cost maps produced by this technique are more accurate than even those produced from hand-trained techniques that utilize superior data sources.

It is important when using overhead data that the data be aligned with the estimated position of the robot. Even slight misregistration can significantly hinder the performance of algorithms such as ours that are sensitive to such errors. An advantage of using an online Bayesian linear regression model is the ability to detect and correct such misalignments with relative ease.

When predicting a new traversal cost c_j, the model creates a predictive distribution $p(c)$ with a mean μ_p and a variance σ_p^2. Evaluating the predictive distribution at the traversal cost c_i of a training example gives the probability of having seen that traversal cost given its corresponding feature vector. We can use the probability of having seen all of our data, $p(\tilde{c}_1, \ldots, \tilde{c}_n)$, to detect map misalignments between overhead data sources and estimated vehicle positions by searching through a space of potential alignments for the one that maximizes the probability of the data.

Since $p(\tilde{c}_1, \ldots \tilde{c}_n)$ can be computed via the chain rule as the product of the predictive distributions evaluated at every $\tilde{c}_i$ used for training, the log data probability is the cumulative sum of $-\log \sigma_p - (y - \mu_p)^2/\sigma_p^2$ for every predictive distribution. After all examples have been received, we compute the average log data probability over all training examples and use this to compare against other alignments. As shown in the following section, correcting such misalignment produces traversal cost maps with better defined obstacles that more accurately reflect the true environment.

Note that for the results in the following section, we chose the hyper-parameter for noise variance σ_l^2 with ML-II and chose an isotropic Gaussian with high variance for the prior on β based on observations from previous robot traversals (Gelman et al., 2004).

4. RESULTS

When operating with overhead data, our algorithm will be referred to as OOLL (overhead on-line learning) and when operating with far-range sensor data, our algorithm will be referred to as FROLL (far-range on-line learning). All imagery data were gathered from satellite on average several months prior to traversal and all elevation data were gathered by surveying from helicopter. Both OOLL and FROLL were run on a 1.8 GHz processor with 2 GB of memory. Because of the large amount of aerial data potentially required by OOLL, we implemented a paging system so that only currently relevant areas of overhead data are kept in memory.

4.1. Field Test Results

The algorithm was tested with both overhead data and far-range sensor data in real-time on-board our

Figure 7. Comparison of paths executed by our robot for shown course when using only on-board perception (in solid red) and with OOLL (in dashed blue) and FROLL (in dotted cyan) used in real-time on-board the robot. Course started at the top right and ended at the bottom left.

unmanned ground vehicle to measure its impact on navigation performance. The test environments contained a large variety of vegetation, various-sized dirt roads (often leading through narrow passages in dense vegetation), hills, and ditches. The vehicle traversed series of courses defined by series of waypoints by using only its on-board perception system for navigation. It then traversed the same courses with the help of OOLL, first with 40 cm resolution overhead imagery and elevation data to supplement the on-board perception system with traversal cost updates computed within a 75 m radius once every 2 s and then with FROLL used to interpret and make predictions from far-range sensor data every 1 s. The algorithm was initialized for each course with no prior training (see Figures 7 and 8 for sample results). Notice how in Figure 7 the OOLL path shows how the robot learned to avoid the dense area of trees after its initial encounter with the area: it immediately chose to follow the road to the goal. The FROLL path shows how the robot chose a similar path to the baseline system but was able to give up on dead ends quicker and was able to avoid the large detour at the end of the course due to its extended perception range.

As shown in Table I, our algorithm allowed the vehicle to complete the courses using OOLL in 26.94% less time while traversing 7.38% less distance and with FROLL in 34.47% less time while traversing 13.26% less distance. Additionally, while we were forced to manually intervene during the tests with only the perception system in order to correct the vehicle's heading when it became trapped in heavy vegetation and could not escape on its own, no manual interventions were necessary when using our algorithm.

While it appears from the shown statistics that FROLL overall resulted in more effective paths than OOLL, we found that with OOLL, the vehicle chose to drive further distances on more preferable terrain in order to avoid difficult or dense areas that presented a larger possibility of damage to the sensors or the need for human intervention. Such an instance can be seen in Figure 8(a).

Figure 8(b) shows a situation where OOLL can be most useful. Since each traverse began with an untrained algorithm, the OOLL course was given an additional waypoint at the right of the image in order to give it an opportunity to train on a small sample of the environment. As seen in Figure 8(d), this small amount of training allowed it to identify the wall of trees that heavily hindered the progress of the vehicle in the traverses using only the perception system or FROLL.

Both techniques not only improved the quality of the path chosen by the vehicle but also allowed higher speed navigation by increasing the time the vehicle had to react to upcoming obstacles and identifying safer terrain such as roads.

4.2. Field Test Data Post-Processing Results

OOLL was also used to produce a priori traversal cost maps for a multi-kilometer course over a large area of complex terrain with heavy vegetation and elevation obstacles defined by a series of GPS waypoints (see Figure 9). The algorithm was trained for about 7 min using two types of overhead imagery data by driving the vehicle by remote control at about 5 m/s through the training course outlined by the red box in Figure 9(a). The trained algorithm was then used offline to generate a traversal cost map and plan an initial path through the much larger course. Closeups of generated traversal cost maps and resulting planned paths are shown. For comparison, we also included the resulting path from a traversal cost map generated by a supervised learning algorithm with human-picked examples from

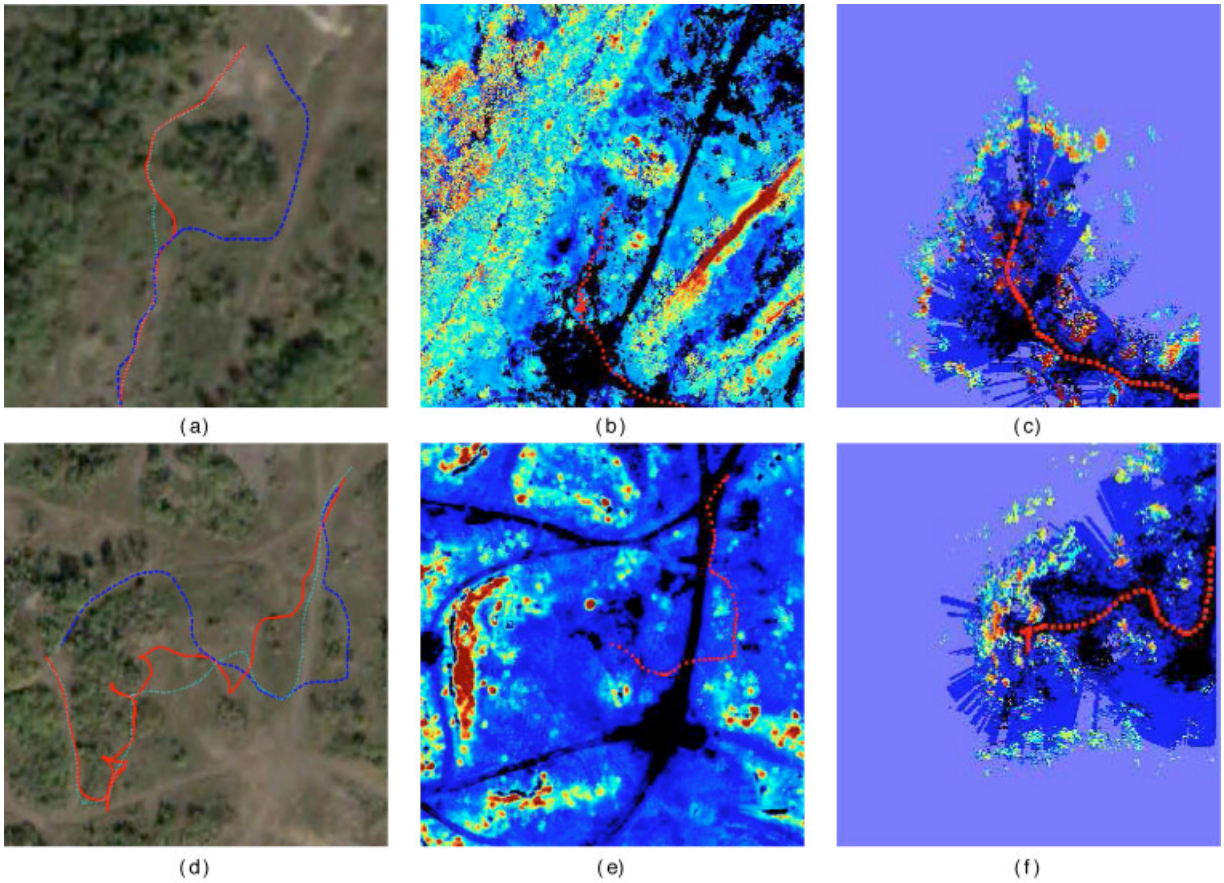


Figure 8. Comparison of paths executed for shown situations when using only on-board perception (in solid red) and with OOLL (in dashed blue) and FROLL (in dotted cyan) are shown in (a) and (d). In (a) the course started at the bottom and ended at the top and in (d) the course started at the top right and ended at the left. Predictions of terrain traversal costs for the environment by our algorithm at the times the vehicle chose to avoid the large obstacles in front of it are shown for OOLL in (b) and (e) and for FROLL in (c) and (f). Traversal costs are color-scaled for improved visibility. Blue and red correspond to lowest and highest traversal costs areas, respectively, with roads appearing in black. In (a) the OOLL path chose to travel slightly further on the road in order to avoid the more difficult passage to the left while the FROLL path was able to detect the opening to the left much sooner than the baseline path. In (b) OOLL helped the vehicle avoid the area of the dense trees by executing a path that is 42.89% shorter in 72.62% less time.

the actual course and features generated from both overhead imagery and high-density elevation data (see Table II for a description of data sources) (Silver et al., 2006).

We evaluated the performance of OOLL against the human-trained technique by accumulating all the traversal costs generated by the vehicle's on-board perception system during a traversal of the

Table I. Statistics for course traversals with and without online learning algorithm

	Without algorithm	With OOLL	With FROLL
Total Traversal time (s)	1369.86	1000.82	897.61
Total distance traveled (m)	1815.71	1681.73	1574.91
Average speed (m/s)	1.33	1.68	1.75
No. of interventions	1	0	0

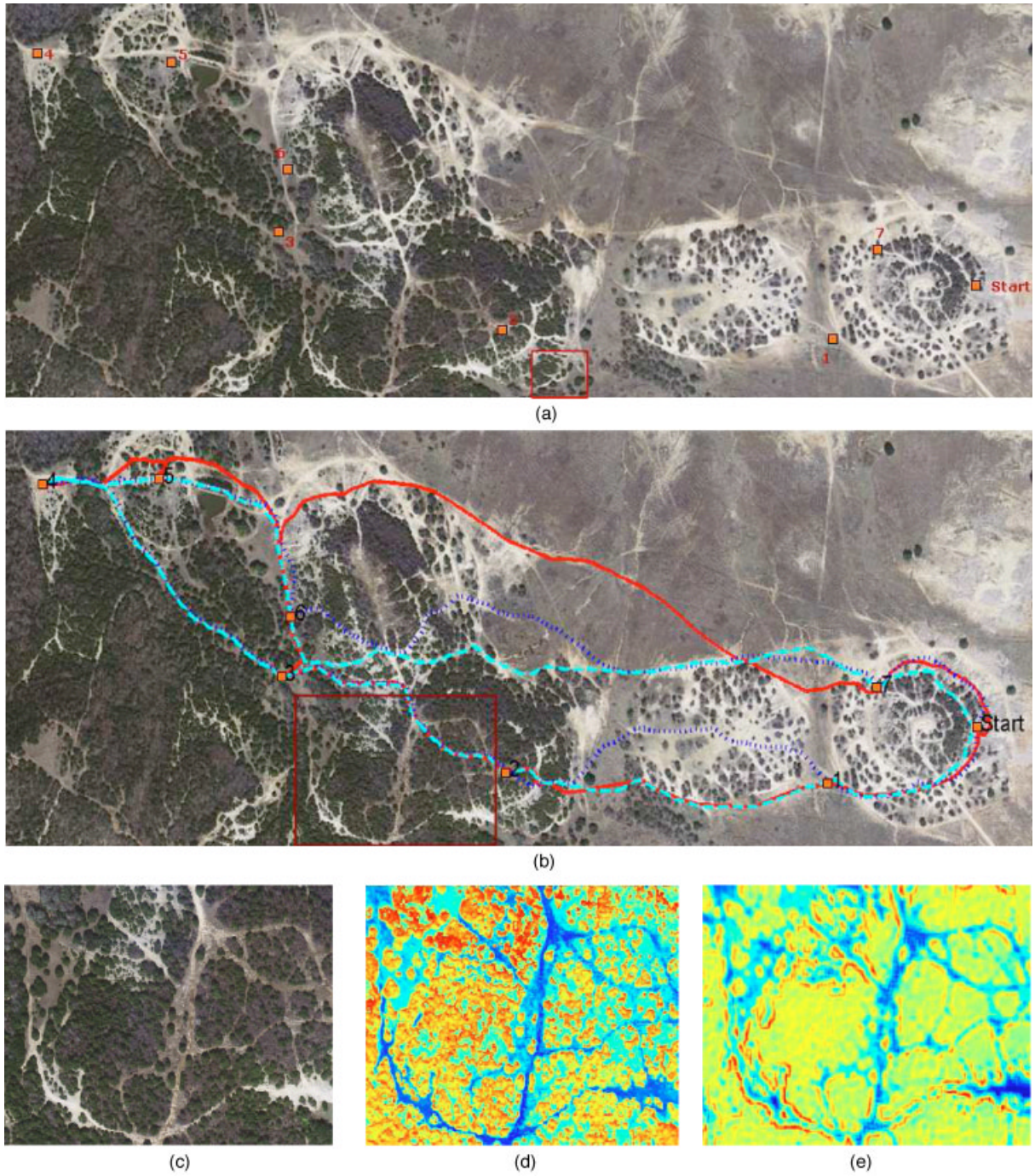


Figure 9. Circular course with the GPS waypoints for which a priori paths were planned is shown in (a). OOLL was trained during a short traversal of the training course outlined in the red box. Shown area is 2000×750 m^2. A priori paths generated by a human-trained algorithm (solid red), OOLL using color imagery data (dashed cyan), and OOLL using black and white imagery data (dotted blue) are shown in (b). Traversal cost maps produced by OOLL for the closeup area in (c) using overhead color imagery and black and white imagery are shown in (d) and (e), respectively. See Table II for description of data sources. Traversal costs are color scaled for improved visibility where blue and red correspond to lowest and highest traversal cost areas, respectively.

Table II. Types of overhead data used by overhead online learning (OOLL) and hand-trained algorithms used to produce prior cost maps.

Algorithm	Data used	Resolution
OOLL (color)	Color imagery	0.35 m
OOLL (B & W)	Terraserver B & W imagery	1.0 m
Human-supervised	Color imagery Elevation	0.35 m <0.2 m

course shown in Figure 9 and comparing those costs to the estimates from each of the generated prior cost maps. The average absolute error in traversal cost (on a log scale as described earlier) for each method is shown in Figure 10 as a function of training time. This result shows that OOLL is competitive with respect to the human-trained algorithm using only imagery data after only a short period of training. However, it should be pointed out that maintaining a tight correspondence from traversal costs assigned by the human-trained algorithm to those assigned by the perception system was difficult to strictly enforce. This highlights another advantage of the online learning approach over a human-trained approach: by relieving the need for manual manipulations of traversal cost assignment strategies, the entire system is more adaptable to changes in both the environment and the perception system.

During postanalysis of this test, we discovered that the overhead imagery data and the estimated position of the vehicle were in fact misaligned by about 1.5 m. While this result shows that our algorithm is robust to such map misalignment, this article also demonstrates how our algorithm can be used to detect such errors in alignment in order to achieve optimal performance.

Performance of FROLL was similarly evaluated by comparing its estimates to all computed perception costs during a course as a function of training time. Only estimates that had not been used for training yet were included in this calculation (so as not to use the training set for testing). The results can be seen in Figure 11. Notice how after only

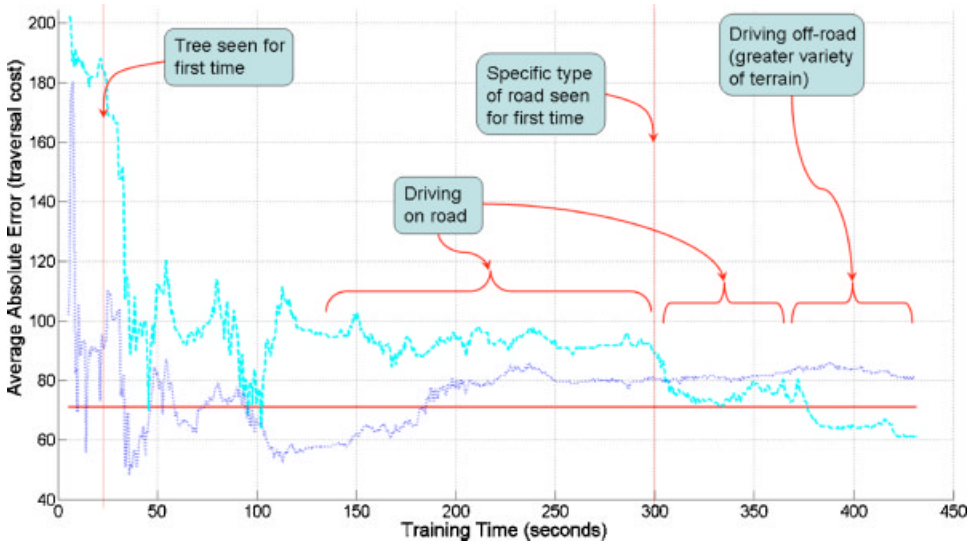


Figure 10. Average absolute error between log scale traversal costs computed by the robot's on-board perception system over the course of a multi-kilometer traverse of terrain and traversal cost estimates computed using three techniques: human-trained supervised learning algorithm using high resolution imagery and elevation data (solid red) and OOLL using only color imagery (dashed cyan) and black and white imagery (dotted blue) as a function of training time by driving in a similar environment. See Table II for a description of data sources. The erratic performance for the first few minutes of training is due to the large effects of new training examples when so few previous data were available. In the case of OOLL with black and white imagery, the initial sample of terrain happened to correspond well to the rest of the course. OOLL training takes longer with color imagery due to a greater number and variety of features.

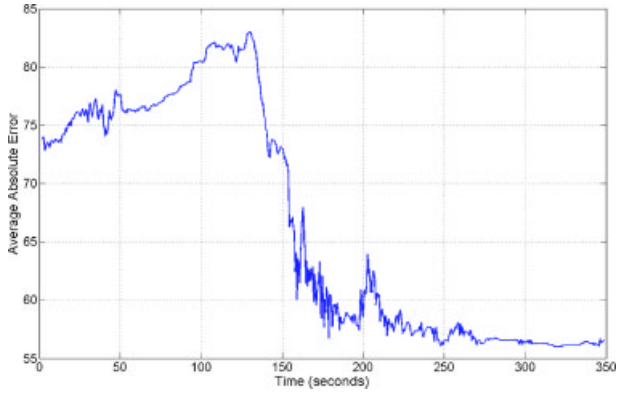


Figure 11. Average absolute error between log scale traversal costs computed by the robot's on-board perception system and traversal cost estimates generated by FROLL as a function of training time.

3 min of training the algorithm has mostly converged to its final predictive performance.

4.3. Offline Map Alignment

We applied our map alignment technique to a manually misaligned log of perception data and overhead imagery features. A brute force search across all potential map alignments in 0.35 m increments in the four cardinal directions detected a misalignment of 3.85 m west and 4.9 m north. Such a search is too computationally expensive to be performed in real-time but was completed in several hours through offline processing. Computed probabilities of observed perception data and the corresponding improvement in traversal cost estimates can be seen in Figure 12. As expected, correcting the misalignment improved the definition of obstacles in the traversal cost maps and resulted in a stronger correspondence with the actual environment, correctly showing that the traveled path is clearly on the road.

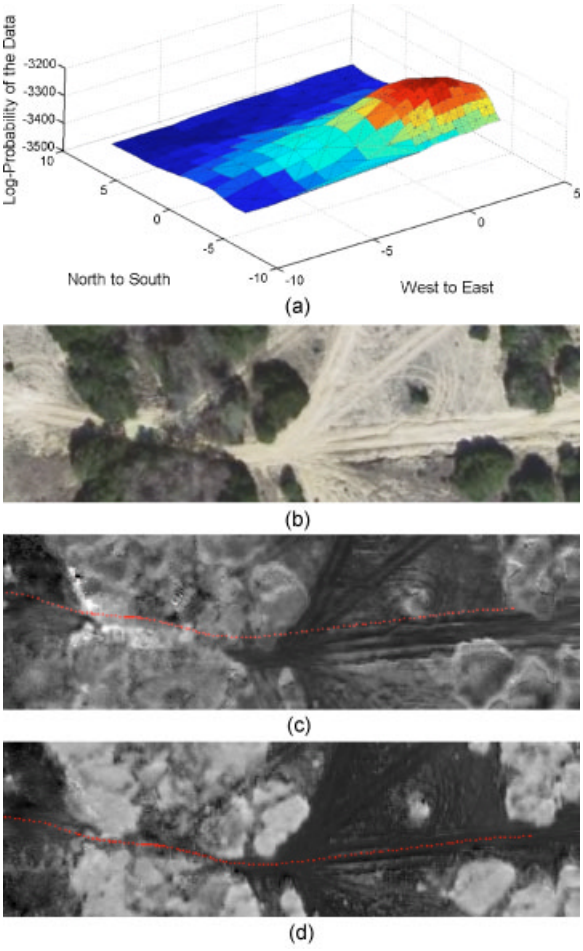


Figure 12. Example of how misalignment between overhead data sources and estimated vehicle position can be detected using our algorithm. Computed log probability of the perception system sensor data encountered over a 12.6×12.6 m^2 search space of alignment shifts is shown in (a). OOLL cost prediction for area shown in (b) before alignment correction and after correcting detected misalignment of 3.85 m west and 4.9 m north appear in (c) and (d), respectively (best alignment is assumed to be that which produces the highest probability of seen perception data). Darker colors in the images correspond to lower traversal costs. The robot's path is shown in red.

4.4. Feature Selection

We applied our feature selection imagery to a set of 33 overhead imagery and elevation data based features. While these features were all relevant to the environment, we assumed that many of them were redundant and therefore selected the 14 most important from the set. Figure 13 depicts the accuracy of cost predictions as a function of training time using this reduced set of features compared to the original set. As expected, the smaller set of selected features resulted in a decreased training time.

5. CONCLUSION

We have proposed a self-supervised online learning algorithm to learn and infer between different types

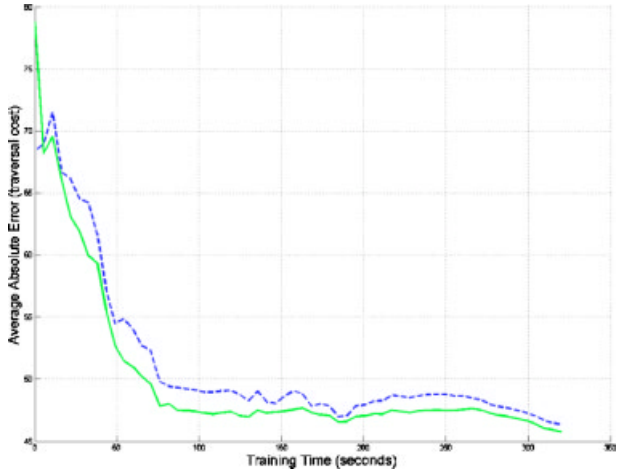


Figure 13. Average absolute error between log scale traversal costs computed by unmanned ground vehicle's onboard perception system over the course of a multikilometer traversal of terrain and traversal cost estimates computed by DOLL before and after feature selection: results of using the full set of 33 features from both imagery and elevation data (dotted blue) and a subset of 14 features selected using the feature selection algorithm (solid green) are shown.

of data sources that vary in density, reliability, and scope. By applying the scoped learning model, we were able to generalize from one type of data source to be able to work with another that may be difficult to generalize to new environments. As a result, we were able to extend the scope of such features to many possible domains.

We showed how the algorithm can be used to improve the navigation capabilities of unmanned ground vehicles by learning in real-time to interpret overhead and far-range sensor data to predict terrain traversal costs generated from an on-board perception system. We demonstrated this approach through field tests on-board a large robot in complex natural environments. Both online and offline results were given to demonstrate several applications of the algorithm. While performance could be hampered because of limitations in the available features or availability of representative training examples, the use of this algorithm was shown to significantly improve the quality of robot navigation performance. Allowing robots to adapt to and improve their performance in diverse environments without human involvement through such techniques greatly expands effectiveness and potential applications of robotic systems.

ACKNOWLEDGMENTS

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government. We would also like to thank all the members of the UPI project team, without whom this work would not be possible.

REFERENCES

Angelova, A., Matthies, L., Helmick, D., Sibley, G., & Perona, P. (2006). Learning to predict slip for ground robots. In Proceedings of the 2006 IEEE International Conference on Robotics and Automation.

Blei, D., Bagnell, J., & McCallum, A. (2002). Learning with scope, with application to information extraction and classification. In Proceedings of the 2002 Conference on Uncertainty in Artificial Intelligence.

Bodta, B., & Camden, R. (2004). Technology readiness level 6 and autonomous mobility. In Proceedings of the SPIE, Volume 5422, Unmanned Ground Vehicle Technology VI.

Duda, R.O., Hart, P.E., & Stork, D.G. (2000). Pattern classification. New York: Wiley-Interscience.

Gelman, A., Carlin, J.B., Stern, H.S., & Rubin, D.B. (2004). Bayesian data analysis. London: Chapman and Hall/CRC.

Goldberg, S., Maimone, M., & Matthies, L. (2002). Stereo vision and rover navigation software for planetary exploration. In Proceedings of the IEEE Aerospace Conference.

Jordan, M.I., & Weiss, Y. (Eds.) (2002). Graphical models: Probabilistic inference. Cambridge, MA: MIT Press.

Kakade, S.M., & Ng, A.Y. (2005). Online bounds for Bayesian algorithms. In Saul, L.K., Weiss, Y., & Bottou, L. (Eds.), Advances in neural information processing systems 17 (pp. 641–648). Cambridge, MA: MIT Press.

Kelly, A., Stentz, A., Amidi, O., Bode, M., Bradley, D., Diaz-Calderon, A., et al. (2006). Toward reliable off road autonomous vehicles operating in challenging environments, International Journal of Robotics Research 25(5/6), 449–484.

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

LeCunn, Y., Muller, U., Ben, J., Cosatto, E., & Flepp, B. (2005). Off-road obstacle avoidance through end-to-end learning. In Proceedings of the Neural Information Processing Systems.

Lieb, D., Lookingbill, A., & Thrun, S. (2005a). Adaptive road following using self-supervised learning and reverse optical flow. In Proceedings of the Robotics Science and Systems.

Lieb, D., Lookingbill, A., & Thrun, S. (2005b). Adaptive road following using self-supervised learning and reverse optical flow. In Proceedings of Robotics: Science and Systems.

Michels, J., Saxena, A., & Ng, A.Y. (2006). High speed obstacle avoidance using monocular vision and reinforcement learning. In Proceedings of the International Conference on Machine Learning.

Minka, T.P. (2001). A family of algorithms for approximate Bayesian inference. Unpublished Ph.D. thesis, Massachusetts Institute of Technology Media Lab, Cambridge, MA.

Ng, A.Y., & Jordan, M.I. (2001). Convergence rates of the voting Gibbs classifier, with application to Bayesian feature selection. In Proceedings of the International Conference on Machine Learning.

Silver, D., Sofman, B., Bagnell, J.A., Vandapel, N., & Stentz, A. (2006). Experimental analysis of overhead data processing to support long range navigation. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems.

Stentz, A. (1994). Optimal and efficient path planning for partially-known environments. In Proceedings of the 1994 IEEE Conference on Robotics and Automation.

Tipping, M.E. (2003). Bayesian inference: An introduction to principles and practice in machine learning. In Proceedings of the Advanced Lectures on Machine Learning.