High Performance Outdoor Navigation from Overhead Data using Imitation Learning

David Silver, J. Andrew Bagnell, Anthony Stentz
Robotics Institute, Carnegie Mellon University
Pittsburgh, Pennsylvania USA

Abstract— **High performance, long-distance autonomous navigation is a central problem for field robotics. Efficient navigation relies not only upon intelligent onboard systems for perception and planning, but also the effective use of prior maps and knowledge. While the availability and quality of low cost, high resolution satellite and aerial terrain data continues to rapidly improve, automated interpretation appropriate for robot planning and navigation remains difficult. Recently, a class of machine learning techniques have been developed that rely upon expert human demonstration to develop a function mapping overhead data to traversal cost. These algorithms choose the cost function so that planner behavior mimics an expert's demonstration as closely as possible. In this work, we extend these methods to automate interpretation of overhead data. We address key challenges, including interpolation-based planners, non-linear approximation techniques, and imperfect expert demonstration, necessary to apply these methods for learning to search for effective terrain interpretations. We validate our approach on a large scale outdoor robot during over 300 kilometers of autonomous traversal through complex natural environments.**

I. Introduction

Recent competitions have served to demonstrate both the growing popularity and promise of mobile robotics. Although autonomous navigation has been successfully demonstrated over long distances through on-road environments, long distance off-road navigation remains a challenge. The varying and complex nature of off-road terrain, along with different forms of natural and man made obstacles, contribute to the difficulty of this problem.

Although autonomous off-road navigation can be achieved solely through a vehicle's onboard perception system, both the safety and efficiency of a robotic system are greatly enhanced by prior knowledge of its environment. Such knowledge allows high risk sections of terrain to be avoided, and low risk sections to be more heavily utilized.

Overhead terrain data are a popular source of prior environmental knowledge, especially if the vehicle has not previously encountered a specific site. Overhead sources include aerial or satellite imagery, digital elevation maps, and even 3-D LiDAR scans. Much of this data is freely available at lower resolutions, and is commercially available at increasing resolution.

Techniques for processing such data have focused primarily on processing for human consumption. The challenge in using overhead data for autonomous navigation is to interpret the data in such a way as to be useful for a vehicle's planning and navigation system. This paper proposes the use of imitation learning to train a system to automatically interpret overhead data for use within an autonomous vehicle planning system. The proposed approach is based on the Maximum Margin Planning (MMP) [1] framework, and makes use of expert provided examples of how to navigate using the provided data.

The next section presents previous work in off-road navigation from prior data. Section III presents the MMP framework, and Section IV develops its application to this context. Results are presented in Section V, and conclusions in Section VI.

II. Outdoor Navigation from Overhead Data

At its most abstract level, the outdoor navigation problem in mobile robotics involves a robot driving itself from a start waypoint to a goal waypoint. The robot may or may not have prior knowledge of the environment, and may or may not be able to gather more environmental knowledge during its traverse. In either case, there exists some form of terrain model. For outdoor navigation, this is often a grid representation of the environment, with a set of environmental features at each grid cell.

The systems we consider in this work rely upon a *planning system* to make navigation decisions. This system is responsible for finding an optimal path that will lead the robot to its goal. The naturally raises the question of how "optimal" will be defined: is it the safest path, the fastest path, the minimum energy path, or the most stealthy path? Depending on the context, these are all valid answers; combinations of the above are even more likely to be desired. In practice, planners typically function by choosing the path with the minimum score according to some scalar function of the environment that approximates those metrics. Therefore, for a chosen metric, the robot's terrain model must be transformed into a single number for every planner state: that state's traversal cost. A planner can then determine the optimal path by minimizing the cumulative cost the robot expects to experience. Whether or not this path achieves the designer's desired behavior depends on how faithfully the cost function, mapping features of the environment to scalar traversal costs, reflects the designer's internal performance metric.

Previous work [2] [3] has demonstrated the effectiveness of overhead data for use in route planning. Overhead imagery, elevation maps, and point clouds are processed into features stored in 2-D grids. These feature maps can then be converted to traversal costs. Figure 1 demonstrates how this approach can lead to more efficient navigation by autonomous vehicles.

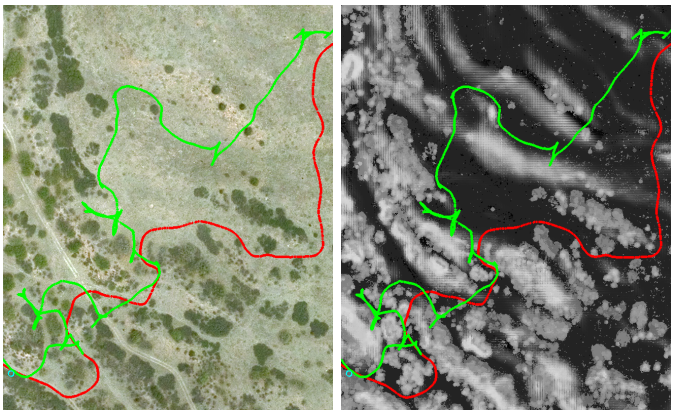

Fig. 1. The path traveled during two runs of an autonomous robot moving from top-right to bottom left. The **green** run had no prior map, and made several poor decisions that required backtracking. The **red** run had the prior costmap shown at right, and took a more efficient route. Brighter pixels indicate higher cost.

Despite good results, previous work has depended on hand tuned cost functions. That is, an engineer manually determined both the form and parameters of a function to map overhead features to a single cost value, in an attempt to express a desired planning metric. By far, human engineering is the most common approach to generating cost functions. This is somewhat of a black art, requiring a deep understanding of the features being used, the metric to be optimized, and the behavior of the planner. Making the situation worse, determining a cost function is not necessarily a one time operation. Each environment could have drastically different overhead input data available. One environment may have satellite imagery, another aerial imagery and an elevation map, another may have multispectral imagery, etc. Therefore, an entire family of cost functions is needed, and parameters must be continually retuned.

One approach to simplifying this problem is to transform the original "raw" feature space into a "cost" feature space, where the contribution of each feature to cost is more intuitive. Classifiers are a common example of this approach, where imagery and other features are transformed into classifications such as road, grass, trees, bushes, etc. However, the tradeoff that is faced is a loss of information. Also, while the cost functions themselves may be simpler, the mapping into this new feature space now must be recomputed or retrained when the raw features change, and the total complexity of the mapping to cost may have increased.

Regardless of any simplifications, human engineering of cost functions inevitably involves parameter tuning, usually until the results are correct by visual inspection. Unfortunately, this does not ensure the costs will actually reflect the chosen metric for planning. This can be validated by planning some sample start/goal waypoints to see if the resulting paths are reasonable. Unreasonable plans result in changes to the cost parameters. This process repeats, expert in the loop, until the expert is satisfied. Often times, continual modifications take the form of very specific rules or thresholds, for very specific shortcomings of the current cost function. This can result in a function with poor generalization.

Hidden in this process is a key insight. While experts have difficulty expressing a specific cost function, they are good at evaluating results. That is, an expert can look at the raw data in its human presentable form, and determine which paths are and are not reasonable. It is this ability of experts that will be exploited for learning better cost functions.

III. Maximum Margin Planning for Learning Cost Functions

Imitation Learning has been demonstrated as an effective technique for deriving suitable autonomous behavior from expert examples. Previous work specific to autonomous navigation [4, 5] has demonstrated how to learn mappings from features of a state to actions. However, these techniques do not generalize well to long range decisions, due to the dimensionality of the feature space that would be required to fully encode the such a problem.

A powerful recent idea for how to approach such long range problems is to structure the space of learned policies to be optimal solutions of search, planning or general Markov Decision Problems [1, 6, 7] . The goal of the learning procedure is then to identify a mapping from features of a state to costs such that the resulting minimum cost plans capture well the demonstrated behavior. We build on the approach of Maximum Margin Planning (MMP) [1, 7], which searches for a cost function that makes the human expert choices appear optimal.

In this method, an expert provided optimal example serves as a constraint on cost functions. If an example path is provided as the best path between two waypoints, then any acceptable cost function should cost the example as less then or equal to all other paths between those two waypoints. By providing multiple example paths, the space of possible cost functions can be further constrained.

Our interest in this work is in applying learning techniques to ease the development of cost functions appropriate for outdoor navigation. The approach to imitation learning described above directly connects the notion of learning a cost function to recovering demonstrated behavior. As we introduce new techniques that improve performance in our application, we sketch the derivation of the functional gradient version [8, 9] of MMP below. This approach allows us to adapt existing, off-the-shelf regression techniques to learn the potentially complicated cost function, leading to a modular and simple to implement technique.

A. Single Example

Consider a state space $\mathcal{S}$, and a feature space $\mathcal{F}$ defined over $\mathcal{S}$. That is, for every $x \in \mathcal{S}$, there exists a corresponding feature vector $F_x \in \mathcal{F}$. C is defined as a cost function over the feature space, $C : \mathcal{F} \rightarrow \mathbb{R}^+$. Therefore, the cost of a state x is $C(F_x)$.

A path P is defined as a sequence of states in $\mathcal{S}$ that lead from a start state s to a goal state g. The cost of P is simply the sum of the costs of all states along the path.

Fig. 2. **Left:** Example paths that imply different metrics (From top to bottom: minimize distance traveled, stay on roads, stay near trees) **Right:** Learned costmaps from the corresponding examples (brighter pixels indicate higher cost). Quickbird imagery courtesy of Digital Globe, Inc.

If an example path P_e is provided, then a constraint on cost functions can be defined such that the cost of P_e must be lower cost than any other path from s_e to g_e. The structured maximum margin approach [1] encourages good generalization and prevents trivial solutions (e.g. the everywhere 0 cost function) by augmenting the constraint to includes a margin: i.e. the demonstrated path must be BETTER than another path by some amount. The size of the margin is dependent on the similarity between the two paths. In this context, similarity is defined by how many states the two paths share, encoded in a function L_e. Finding the optimal cost function then involves constrained optimization of an objective functional over C

$$\min \mathcal{O}[C] = \textsc{Reg}(C) \tag{1}$$

subject to the constraints

$$\sum_{x \in \hat{P}} (C(F_x) - L_e(x)) \; - \; \sum_{x \in P_e} (C(F_x)) \geq 0$$

$$\forall \hat{P} \; s.t. \; \hat{P} \neq P_e, \;\; \hat{s} = s_e, \;\; \hat{g} = g_e$$

$$L_e(x) = \begin{cases} 1 & \text{if } x \in P_e \\ 0 & \text{otherwise} \end{cases}$$

where REG is a regularization operator that encourages generalization in the the cost function C.

There are typically an exponentially large (or even infinite) number of constraints, each corresponding to an alternate path. However, it is not necessarily to enumerate these constraints. For every candidate cost function, there is a minimum cost path between two waypoints and at each step it is only necessary to enforce the constraint on this path.

It may not always be possible to achieve all constraints and thus a "slack" penalty is added to account for this. Since the slack variable is tight, we may write an "unconstrained" problem that captures the constraints as penalties:

$$\min \mathcal{O}[C] = \textsc{Reg}(C) + \sum_{x \in P_e} (C(F_x)) \; - \; \min_{\hat{P}} \sum_{x \in \hat{P}} (C(F_x) - L_e(x)) \tag{2}$$

For linear cost functions (and convex regularization) $\mathcal{O}[C]$ is convex, and can be minimized using gradient descent. [1]

Linear cost functions may be insufficient, and using the argument in [8], it can be shown that the (sub-)gradient in the space of cost functions of the objective is given by [2]

$$\bigtriangledown \mathcal{O}_F[C] = \sum_{x \in P_e} \delta_F(F_x) \;\; - \sum_{x \in P_*} \delta_F(F_x) \tag{3}$$

Simply speaking, the functional gradient is positive at values of F that the example path pass through, and negative at values of F that the planned path pass through. The magnitude of the gradient is determined by the frequency of visits. If the example and planned paths agree in their visitation counts at F, the functional gradient is zero at that state. Applying gradient descent in this space of cost functions directly would involve an extreme form of overfitting: defining a (potentially unique) cost at every value of F encountered and involving no generalization. Instead, as in gradient boosting [9] we take a small step in a limited set of "directions" using a hypothesis space of functions mapping features to a real number. The result is that the cost function is of the form:

$$C(F) = \sum \eta_i R_i(F) \tag{4}$$

where R belongs to a chosen family of functions (linear, decision trees, neural networks, etc.) The choice of hypothesis space in turn controls the complexity of the resulting cost function. We must then find at each step the element R_* that maximizes the inner product $\langle - \bigtriangledown \mathcal{O}_F[C], R_* \rangle$ between the functional gradient and elements of the space of functions we are considering. As in [8], we note that maximizing the inner product between the functional gradient and the hypothesis space can be understand as a learning problem:

$$R_* = \arg\max_R \sum_{x \in P_e \cap P_*} \alpha_x y_x R(F_x) \tag{5}$$

$$\alpha_x = | \bigtriangledown \mathcal{O}_{F_x}[C] | \quad y_x = -\text{sgn}(\bigtriangledown \mathcal{O}_{F_x}[C])$$

In this form, it can be seen that finding the projection of the functional gradient essentially involves solving a weighted classification problem. Performing regression instead of classification adds regularization to each projection [8].

Intuitively, the regression targets are positive in places the planned path visits more than the example path, and negative

[1] Technically, sub-gradient descent as the function is non-differentiable

[2] Using δ to signify the dirac delta at the point of evaluation

$C_0 = 1$;
for $i = 1...T$ **do**
  $\mathcal{D} = \emptyset$;
  foreach P_e **do**
    $\mathcal{M} =$ `buildCostmapWithMargin(`$C_{i-1}, s_e, g_e, \mathcal{F}$`);`
    $P_* =$ `planPath(`$s_e, g_e, \mathcal{M}$`);`
    $\mathcal{D} = \mathcal{D} \bigcup \{P_e, -1\}$;
    $\mathcal{D} = \mathcal{D} \bigcup \{P_*, 1\}$;
  end
  $R_i =$ `trainRegressor(`$\mathcal{F}, \mathcal{D}$`);`
  $C_i = C_{i-1} * e^{\eta_i R_i}$;
end

Algorithm 1: The proposed imitation learning algorithm

in places the example path visits more than the planned path. The weights on each regression target are the difference in visitation counts. In places where the visitation counts agree, both the target value and weight are 0. Pseudocode for the final algorithm is presented in Algorithm 1.

Gradient descent can be understood as encouraging functions that are "small" in the l_2 norm. If instead, we consider applying an *exponentiated functional gradient descent* update as described in [7, 10] we encourage functions that are "sparse" in the sense of having many small values and a few potentially large values. Instead of cost functions of the form in (4) this produces functions of the form

$$C(F) = e^{\sum \eta_i R_i(F)} \tag{6}$$

This naturally results in cost functions which map to $\mathbb{R}^+$, without any need for projecting the result of gradient descent into the space of valid cost functions. We believe, and our experimental work demonstrates, that this implicit prior is more natural and effective for problems of outdoor navigation.

B. Multiple Examples

Each example path defines a constraint on the cost function. However, it is rarely the case that a single constraint will sufficiently inform the learner to generalize and produce reasonable planner behavior for multiple situations. For this reason, it is often desireable to have multiple example paths.

Every example path will produce its own functional gradient as in (3). When learning with multiple paths, these individual gradients can either be approximated and followed one at a time, or summed up and approximated as a batch. To minimize training time, we have chosen the later approach.

One remaining issue is that of relative weighting between the individual gradients. If they are simply combined as is, then longer paths will have a greater contribution, and therefore a greater importance when balancing all constraints. Alternatively, each individual gradient can be normalized, giving every path equal weight. Again, we have chosen the latter approach. The intuition behind this is that every path is a single constraint, and all constraints should be equally important. However, this issue has not been thoroughly explored.

IV. Applied Imitation Learning on Overhead Data

At a high level, the application of this approach to overhead data is straightforward. $\mathcal{S}$ is $SE(2)$, with features extracted from overhead data at each 2-D cell. Example paths are drawn over this space by a domain expert. Paths can also be provided by driving over the actual terrain, when access to the terrain is possible before the learned map is required. However, there are several practical problems encountered during application of this algorithm that require solutions.

A. Adaptation to Different Planners

The algorithm suggested in the previous section requires determining the visitation counts for each cell along both the example and the current planned path. For some planners this is straightforward; a simple A* planner visits each cell on the path once. For other planners, it is not so simple. Since the goal of this work is to learn the proper cost function *for a specific planner*, planner details must be taken into account.

Many planners apply a configuration space expansion to an input cost map before planning, to account for the dimensions of the vehicle. When such an expansion does take place, it must also be taken into account by the learner. For example, if the planner takes the average cost of all cells within a window around the current cell, then the visitation count of all the cells within the same window must be incremented, not just the current cell. If the expansion applies different weightings to different cells within a window, then this weighting must also be applied when incrementing the visitation counts.

More complicated planners may also require non-unit changes to the visitation counts, regardless of an expansion. For example the interpolated A* algorithm [11] used in this work generates paths that are interpolated across cell boundaries. Therefore, the distance traveled through a cell may be anywhere in the interval $(0, \sqrt{2}]$ (in cells) As the cost of traveling through a cell under this scheme is proportional to the distance traveled through the cell, the visitation counts must also be incremented proportional to the distance traveled through the cell.

B. Regressor Choice

The choice of regressor for approximating the functional gradient can have a large impact on performance. Simple regressors may generalize better, and complex regressors may learn more complicated cost functions. This tradeoff must be effectively balanced. As in a classification task, one way to accomplish this is to observe performance on an independent validation set of example paths.

Linear regressors offer multiple advantages in this context. Aside from simplicity and good generalization, a weighted combination of linear functions simply produces another linear function. This provides a significant decrease in processing time for repeated cost evaluations.

The MMPBoost [8] algorithm can also be applied. This algorithm suggests using linear regressors, and then occasionally performing a nonlinear regression. However, instead of adding this nonlinear regressor directly to the cost function,

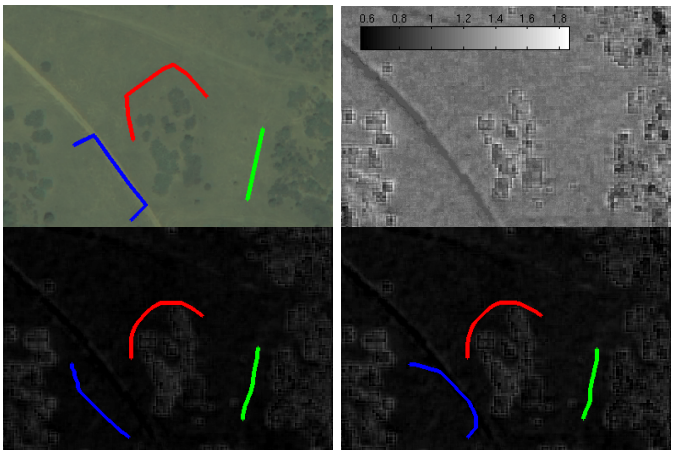

Fig. 3. **Top Left:**The red path is an unachievable example path, as it will be less expensive under any cost function to cut more directly accross the grass. **Bottom Left:** With standard weighting, the unachievable example forces down the cost of grass, and prevents the blue example from being achieved. **Bottom Right:** Balanced weighting prevents this bias, and the blue path is achieved **Top Right:** The ratio of the balanced to the standard costmap. The cost of grass is higher in the balanced map.

it is added as a new feature for future linear regressors. In this way, future iterations can tune the contribution of these occasional nonlinear steps, and the ammount of nonlinearity in the final cost function is tightly controlled.

C. Unachievable Example Paths

The derivations in Section III operate under the assumption that the example paths are either optimal or are just slightly sub-optimal, with the error being taken into the slack variables in the optimization problem. In practice, it is often possible to generate an example path such that no consistent cost metric will make the example optimal.

Figure 3 provides a simple example of such a case. The red example path takes a very wide berth around some trees. All the terrain the path travels over is essentially the same, and so will have the same cost. If the spacing the path gives the trees is greater than the expansion of the planner, then no cost function will ever consider that example optimal; it will always be better to cut more directly across the grass. In practice, such a case occurs in a small way on almost all human provided examples. People rarely draw or drive perfect straight lines or curves, and therefore generate examples that are a few percent longer than necessary. Further, a limited set of features and planner limitations mean that the path a human truly considers to be optimal may not be achievable using the planning system and features available to learn with.

It is interesting to note what happens to the functional gradient with an unachievable example. Imagine an example path that only travels through terrain with an identical feature vector F'. Under any cost function, the planned path will be the shortest path from start to goal. But what if the example path takes a longer route? The functional gradient will then be positive at F', as the example visitation count is higher than the planned. Therfore, the optimization will try to lower the cost of F'. At the next epoch, neither path will have changed. So unachievable paths result in components of the functional gradient that continually try to lower costs. These components can counteract the desires of other, achievable paths, resulting in worse cost functions (see Figure 3).

This effect can be counteracted with a small modification to the learning procedure of the functional gradient. Simplifying notation, define the negative gradient vector as

$$-\bigtriangledown \mathcal{O}_F[C] = U_* - U_e$$

Where U_* and U_e are the visitation counts of the planned and example paths, respectively.

Now consider a new gradient vector that looks at the feature counts normalized by the length of the corresponding path

$$G_{avg} = \frac{U_*}{|P_*|} - \frac{U_e}{|P_e|} \tag{7}$$

In the unachievable case described above, this new gradient would be zero instead of negative. That is, it would be satisfied with the function as is. It can be shown that this new gradient still correlates with the original gradient in other cases:

$$\begin{aligned} \langle -\bigtriangledown \mathcal{O}_F[C], G_{avg} \rangle &> 0 \\ \langle U_* - U_e, \frac{U_*}{|P_*|} - \frac{U_e}{|P_e|} \rangle &> 0 \\ \frac{\langle U_*, U_* \rangle - \langle U_*, U_e \rangle}{|P_*|} + \frac{\langle U_e, U_e \rangle - \langle U_*, U_e \rangle}{|P_e|} &> 0 \end{aligned}$$

The term $\langle U_*, U_e \rangle$ can be understood as related to the similarity in visitation counts between the two paths. If the two paths visit terrain with different features, this term will be small, and the inequality will hold. If the paths visit the exact same features ($U_* = U_e$) then the inequality does not hold. This is the situation discussed above, where the gradient is zero instead of negative.

In terms of the approximation to the gradient, this modification suggest performing a balanced classification or regression on the visitation counts as opposed to the standard, weighted one. As long as the classifier can still separate between the positive and negative classes, the balanced result will point in the same direction as the standard one. When the classifier can no longer distinguish between the classes, the gradient will tend more to towards zero, as opposed to moving in the direction of the more populous class (see Figure 3). This balancing also accomplishes the functional gradient normalization between paths described in section III-B.

D. Alternate Examples

When example paths are unachievable or inconsistent, it is often by a small amount, due to the inherent noise in a human provided example. One way to address this is to slightly redefine the constraints on cost functions. Instead of considering an example path as indicative of the exact optimal path, it can be considered instead as the approximately optimal path. That is, instead of trying to enforce the constraint that no path is cheaper than the example path, enforce the constraint that no path is cheaper than the cheapest path that exists

completely within a corridor around the example path. With this constraint, the new objective functional becomes

$$\mathcal{O}[C] = \min_{\hat{P}_e} \sum_{x \in \hat{P}_e} C(F_x) \quad - \quad \min_{\hat{P}} \sum_{x \in \hat{P}} C(F_x) - L_e(x) \quad (8)$$

Where $\hat{P}_e$ is the set of all paths within the example corridor. The result of this new set of constraints is that the gradient is not affected by details smaller than the size of the corridor. In effect, this acts as a smoother that can reduce noise in the examples. However, if the chosen corridor size is too large, informative training information can be lost.

One downside of this new formulation is that the objective functional is no longer convex (due to the first $\min$ term). It is certainly possible to construct cases with a single path where this new formulation would converge to a very poor cost function. However, empirical observation has indicated that as long as more than a few example corridors are used, all the local minima achieved are quite satisfactory.

More complicated versions of this alternative formulation are also possible. For instance, instead of a fixed width corridor along the example path, a specified variable width corridor could be used. This would allow example paths with high importance at some sections (at pinch points) and low importance elsewhere. Another version of this formulation would involve multiple disjoint corridors. This could be used if an expert believed there were two different but equally desirable "best" paths from start to goal.

V. Experimental Results

A. Algorithm Verification

In order to understand the performance of this approach, a series of offline experiments was performed on real overhead data. The dataset consisted of Quickbird satellite imagery and a 3-D LiDAR scan with approximately 3 points per m^2. The metric used to evaluate the results of these experiments is the average *cost ratio* over all paths. The cost ratio is the cost of the example path over the cost of the planned path, and is proportional to the normalized value of the objective functional. As the cost function comes closer to meeting all constraints, this ratio approaches 1. All experiments were performed using linear regressors, unless otherwise stated.

1) Simulated Examples: In order to first verify the algorithm under ideal conditions, tests were run on simulated examples. A known cost function was used to generate a cost map, from which paths between random waypoints were planned. Different numbers of these paths were then used as input for imitation learning, and the performance measured on an independent test set of paths generated in the same manner.

Figure 4 shows the results using both the balanced and standard weighting schemes. As the number of training paths is increased, the test set performance continues to improve. Each input path further constrains the space of possible cost functions, bringing the learned function closer to the desired one. However, there are diminishing returns as additional

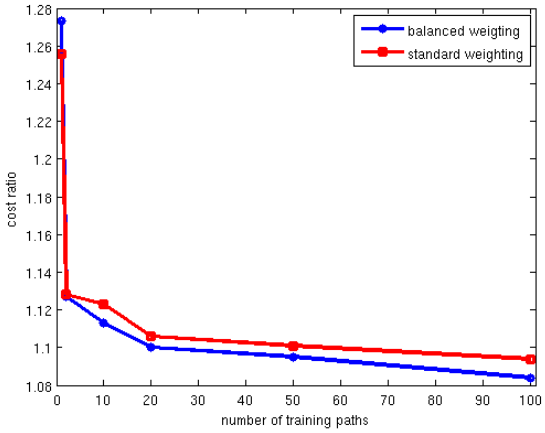

Fig. 4. Learning with simulated example paths. Test set performance is shown as a function of number of input paths.

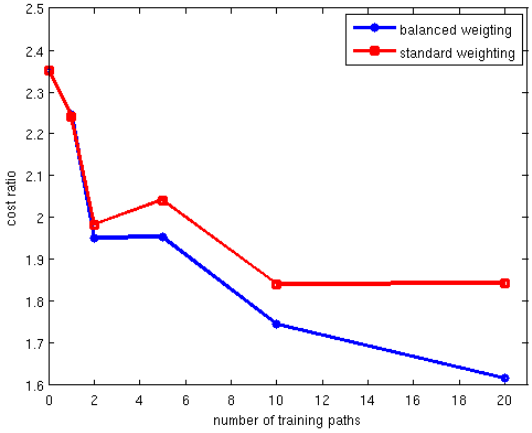

Fig. 5. Learning with expert drawn example paths. Test set performance is shown as a function of number of input paths.

paths overlap to some degree in their constraints. Finally, the performance of the balanced and standard weighting schemes is similar. Since all paths for this experiment were generated by a planner, they are by definition optimal under some metric.

2) Expert Drawn Examples: Next, experiments were performed with expert drawn examples. Figure 5 shows the results of an experiment of the same form as that performed with simulated examples. Again, the test set cost ratio decreases as the number of training examples increases. However, with real examples there is a significant difference between the two weighting schemes. The balanced weighting scheme achieved significantly better performance than the standard one. This difference in performance is further shown in Figure 6. Both the training and test performance are better with the balanced weighting scheme. Further, with the standard weighting scheme, the distribution of costs is shifted lower, due to the negative contribution to the gradient of unachievable paths.

To demonstrate the differences in performance when using different regressors, cost functions were trained using linear

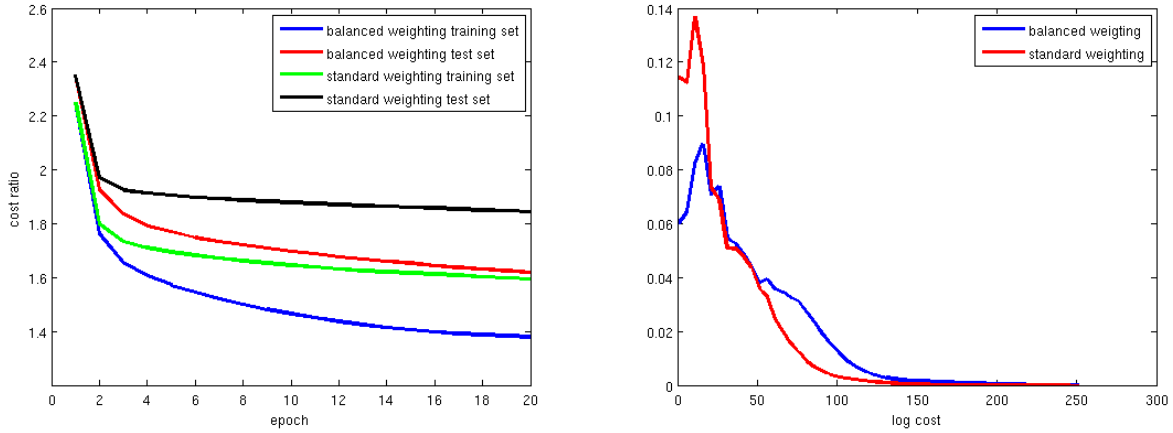

Fig. 6. **Left:** The training and test cost ratio, under both weighting schemes, as a function of the number of epochs of training. This test was run with 20 training and 20 test paths. **Right:** histogram of the costs produced by both weighting schemes. The standard weighting scheme produces lower costs.

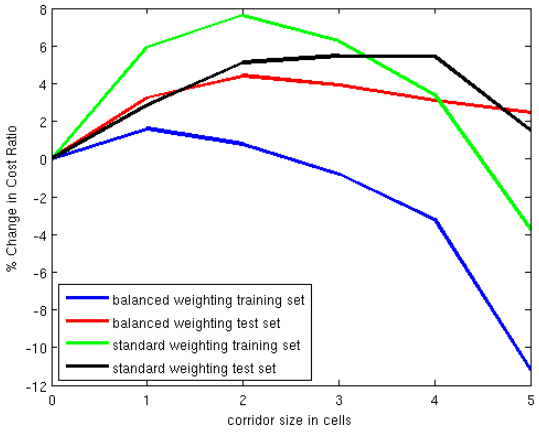

Fig. 7. Improvement in final cost ratio as a function of the corridor size

regressors, and simple regression trees (maximum 4 leaf nodes). The training/test cost ratio with linear regression was 1.23/1.35, and 1.13/1.61 with regression trees. This demonstrates the lack of generalization that can occur with even simple nonlinear regressors.

3) Corridor Constraints: A seres of experiments were performed to determine the effect of using corridor constraints. Figure 7 shows the results as a function of the corridor size in cells. Small corridors provide an improvement over no corridor. However, as the corridor gets too large, this improvement disappears; large corridors essentially over-smooth the examples. The improvement due to using corridor constraints is larger when using the standard weighting scheme, as the balanced scheme is more robust to noise in the examples.

B. Offline Validation

Next, experiments were performed in order to compare the performance of learned costmaps with engineered ones. A cost map was trained off of satellite imagery for an approximately 60 km^2 size area. An engineered costmap had been previously produced for this same area to support the DARPA UPI program (see section V-C). This map was produced by performing a supervised classification of the imagery, and then determining a cost for each class [3]. A subset of both maps is shown in Figure 8.

The two maps were compared using a validation set of paths generated by a UPI team member not directly involved in the development of overhead costing. The validation cost ratio was 2.23 with the engineered map, and 1.63 with the learned map.

C. Online Validation

The learning system's applicability to actual autonomous navigation was validated through the DARPA UPI program. These tests involved a six wheeled, skid steer autonomous vehicle. The vehicle is equipped with laser range finders and cameras for on-board perception. The perception system produces costs that are merged with prior costs into a single map. Due to the dynamic nature of this fused cost map, the Field D* [11] algorithm is used for real-time global planning. The full UPI Autonomy system is described in [12].

The entire UPI system was demonstrated during three large field tests during the past year. Each of these tests consisted of the vehicle autonomously traversing a series of courses, with each course defined as a set of widely spaced waypoints. The tests took place at different locations, each with highly varying local terrain characteristics.

Previous to these latest tests, prior maps for the vehicle were generated as described in [3]. Satellite Imagery and aerial fixed wing LiDAR scans were used as input to multiple feature extractors and classifiers. These features were then fed into a hand tuned cost function. During these most recent three tests, example paths were used to train prior cost maps from the available overhead features. The largest map covered an area of over 200 km^2. Overall, learned prior maps were used during over 300 km of sponsor monitored autonomous traverse.

In addition, two direct online comparisons were performed.

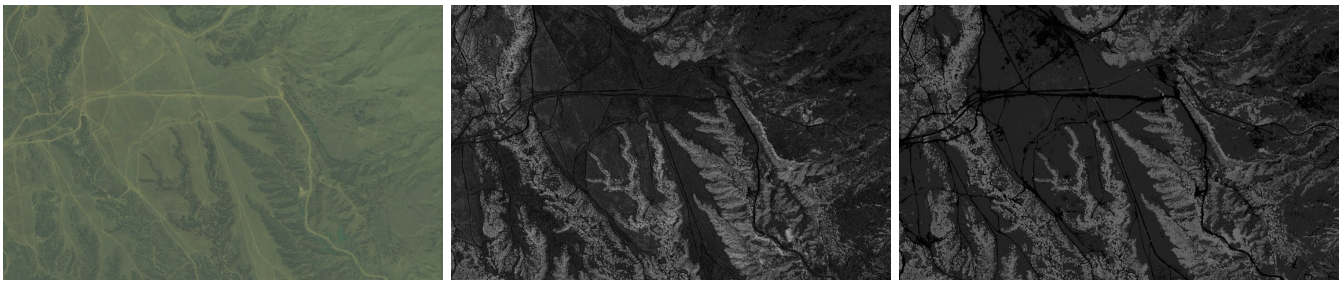

Fig. 8. A 10 km^2 section of a UPI test site. From left to right: Quickbird imagery, MMP Cost, and Engineered Cost

Experiment	Total Net Distance(km)	Avg. Speed(m/s)	Total Cost Incurred	Max Cost Incurred
Experiment 1 Learned	6.63	2.59	11108	23.6
Experiment 1 Engineered	6.49	2.38	14385	264.5
Experiment 2 Learned	6.01	2.32	17942	100.2
Experiment 2 Engineered	5.81	2.23	21220	517.9
Experiment 2 No Prior	6.19	1.65	26693	224.9

TABLE I

These two tests were performed several months apart, at different test sites. During these experiments, the same course was run multiple times, varying only the prior cost map given to the vehicle. Each run was then scored according to the total cost incurred by the vehicle according to its onboard perception system.

The results of these experiments are shown in Table I. In both experiments, the vehicle traveled farther to complete the same course using MMP trained prior data, and yet incurred less total cost. Over both experiments, with each waypoint to waypoint section considered an independent trial, the improvement in average cost and average speed is statistically significant at the 5% and 10% levels, respectively. This indicates that the terrain the vehicle traversed was on average safer when using the learned prior, according to its own onboard perception system. This normalization by distance traveled is necessary because the learned prior and perception cost functions do not necessarily agree in their unit cost.

The course for Experiment 2 was also run without any prior data; the results are presented for comparison.

VI. Conclusion

This paper addresses the problem of interpreting overhead data for use in long range outdoor navigation. Once provided with examples of how a domain expert would navigate based on the data, the proposed imitation learning approach can learn mappings from raw data to cost that reproduce similar behavior. This approach produces cost functions with less human interaction than hand tuning, and with better performance in both offline and online settings.

Future research will focus more on interactive techniques for imitation learning. By specifically prompting an expert with candidate waypoint locations, it is hoped that more information can be derived from each path, thereby requiring fewer examples. The adaptation of these techniques to an onboard perception system will also be explored.

Acknowledgments

This work was sponsored by the Defense Advanced Research Projects Agency (DARPA) under contract "Unmanned Ground Combat Vehicle - PerceptOR Integration" (contract number MDA972-01-9-0005). 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.

References

[1] N. Ratliff, J. Bagnell, and M. Zinkevich, "Maximum margin planning," in *International Conference on Machine Learning*, July 2006.

[2] N. Vandapel, R. R. Donamukkala, and M. Hebert, "Quality assessment of traversability maps from aerial lidar data for an unmanned ground vehicle," in *Proceedings of the IEEE/JRS International Conference on Intelligent Robots and Systems*, October 2003.

[3] D. Silver, B. Sofman, N. Vandapel, J. A. Bagnell, and A. Stentz, "Experimental analysis of overhead data processing to support long range navigation," in *Proceedings of the IEEE/JRS International Conference on Intelligent Robots and Systems*, October 2006.

[4] D. Pomerleau, "Alvinn: an autonomous land vehicle in a neural network," *Advances in neural information processing systems 1*, pp. 305 – 313, 1989.

[5] C. Sammut, S. Hurst, D. Kedzier, and D. Michie, "Learning to fly," in *Proceedings of the Ninth International Conference on Machine Learning*, 1992, pp. 385–393.

[6] P. Abbeel and A. Ng, "Apprenticeship learning via inverse reinforcement learning," in *International Conference on Machine Learning*, 2004.

[7] J. A. Bagnell, J. Langford, Y. Low, N. Ratliff, and D. Silver, "Learning to search," 2008, manuscript in preparation.

[8] N. Ratliff, D. Bradley, J. Bagnell, and J. Chestnutt, "Boosting structured prediction for imitation learning," in *Advances in Neural Information Processing Systems 19*. Cambridge, MA: MIT Press, 2007.

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

[10] J. A. Bagnell, J. Langford, N. Ratliff, and D. Silver, "The exponentiated functional gradient algorithm for structured prediction problems," in *The Learning Workshop*, 2007.

[11] D. Ferguson and A. Stentz, "Using interpolation to improve path planning: The field d* algorithm," *Journal of Field Robotics*, vol. 23, no. 2, pp. 79–101, February 2006.

[12] A. Stentz, "Autonomous navigation for complex terrain," Carnegie Mellon Robotics Institute Technical Report, manuscript in preparation.