Perceptual Interpretation for Autonomous Navigation through Dynamic Imitation Learning

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

Abstract Achieving high performance autonomous navigation is a central goal of field robotics. Efficient navigation by a mobile robot depends not only on the individual performance of perception and planning systems, but on how well these systems are coupled. When the perception problem is clearly defined, as in well structured environments, this coupling (in the form of a cost function) is also well defined. However, as environments become less structured and more difficult to interpret, more complex cost functions are required, increasing the difficulty of their design. Recently, a class of machine learning techniques has been developed that rely upon expert demonstration to develop a function mapping perceptual data to costs. These algorithms choose the cost function such that the robot's planned behavior mimics an expert's demonstration as closely as possible. In this work, we extend these methods to address the challenge of dynamic and incomplete online perceptual data, as well as noisy and imperfect expert demonstration. We validate our approach on a large scale outdoor robot with hundreds of kilometers of autonomous navigation through complex natural terrains.

1 Introduction

The capability of mobile robots to autonomously navigate through unknown environments continues to advance. Especially in structured environments, the effectiveness of both perception and planning systems have been greatly improved. In such environments, the actions that a robot can and cannot take are generally well defined. It is then the task of a perception system to report these definitions, and of a planning system to indicate a series of actions to achieve a desired goal.

However, as environments become less structured, the difficulty of coupling perception and planning systems increases. Under these conditions, the final output of a perception system cannot be binary: a more analog measure of traversability or desirability is required. For example, in outdoor terrain it may be desirable to avoid tall grass in favor of short grass, but it may also be desirable to avoid a bush in favor

Fig. 1 Left: The Crusher platform is capable of long range autonomous navigation through complex terrain. **Center:** Raw perception data from Crusher, in the form of camera images and colorized LiDAR. **Right:** 2D costs derived from perception data. Brighter pixels indicate higher cost.

of either, and to avoid a tree in favor of anything. The computation of such an analog ordering has a drastic effect on the final performance of a mobile robot, and is often a barrier to truly robust navigation.

This paper proposes the use of imitation learning to derive the proper coupling between a mobile robot's perception and planning systems in order to improve autonomous performance. The presented approach is based on extension of the Maximum Margin Planning (MMP) [12, 13] framework, and makes use of expert demonstration of desired navigation behavior. Existing techniques are adapted to account for the unknown and dynamic nature of online perceptual data, as well as to account for noisy and imperfect demonstration. The application of this approach to the Crusher autonomous platform [18] (Figure 1) is then presented. Experimental results are gathered over hundreds of kilometers of autonomous traverse through complex, natural terrains.

2 Perceptual Interpretation For Autonomous Navigation

The canonical task of an autonomous navigation system is to safely guide a robot from a start to a goal location. Knowledge of the environment comes from a combination of any prior data available, and data collected by the robot's onboard sensors. Due to map resolution limitations and post mapping changes, the environment is typically not fully known before the robot navigates. Since the robot's knowledge of its environment evolves over time, an onboard planning system must continually make decisions geared towards achieving its goal. Depending on the complexity of the navigation task, the overall planning system may consist of multiple individual planners. It is also common for such systems to operate in a discretized state space that allows perceptual data to be associated with each state.

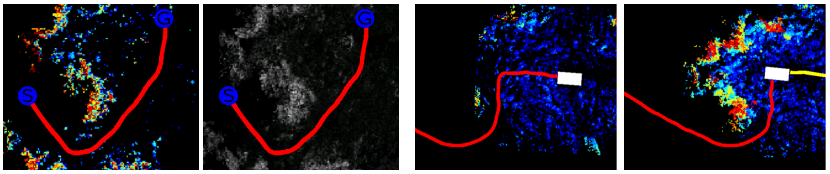
A planning system attempts to produce the optimal sequence of actions that will lead the robot to its goal. This naturally raises the question of how to determine what is "optimal". In well structured environments, this may be well defined: for example, in an indoor environment, the optimal path may be the shortest collision free path to the goal. The perceptual challenge in this scenario is to properly classify sources of collisions; after that, defining optimality is trivial.

However, in complex unstructured environments, defining optimality is more difficult (along with the rest of the perception task). In such environments it is no longer easily definable what is and is not traversable; for example it may or may not be possible to drive over a small bush, or to drive through a large patch of mud without becoming stuck. Traversability must be considered as a probability or some other analog measure. Further, maximizing traversability may not be the only consideration; there might be a desire to minimize energy usage or time taken, maximize the field of view of onboard sensors, or minimize exposure to external sensors. In these scenarios, an analog ordering of preferences and tradeoffs is required, as opposed to a binary classification of passable regions. This in turn creates the requirement for planners that can make use of such orderings. One of the most field proven approaches is to use A* style grid planners, either separately or in conjunction with more continuous local motion planners [7, 15, 17, 18].

The process of converting a perception system's description of an environment to an ordering of preferences is often called costing, with a cost function mapping perceptual data associated with discrete states to costs. In turn, it is then the function of the planning system to attempt to minimize the accrued cost during navigation. By concretely defining relative preferences and tradeoffs, the cost function has a large impact on a mobile robot's behavior. Often, it is chosen in an attempt to force a robot to approximate some intuitive metric (distance traveled, time taken, energy expended, mission risk, etc.) or combination of such. Whether the robot's behavior actually reflects its designed intent is heavily dependent on the mapping from perceptual features to costs. If the perceptual features are sufficiently descriptive of the environment, the cost function can be seen as an encoding of the desired behavior of the robot.

Unfortunately, the more complex the environment and the desired behavior, the more difficult the task of defining the cost function becomes. Human engineering is often the first approach to tackling this problem; it is also potentially quite time consuming. Complex environments necessitate full featured and high dimensional descriptions, often on the order of dozens of features per discrete state. Worse still, there is often not a clear relationship between features and cost. Therefore, engineering a cost function by hand is akin to manually solving a high dimensional optimization problem using local gradient methods. Evaluating each candidate function then requires validation through either actual or simulated robot performance. This tedious process must be repeated whenever the input perceptual features are modified, or incorrect behavior is observed in a novel environment. Despite these drawbacks, manual engineering has been popular for lack of alternatives [4, 5, 7, 17]. Engineering or learning more intuitive features (with respect to costing) simplifies but does not solve this problem (while creating additional work elsewhere in the perception pipeline). Self-supervised approaches that learn to interpret an environment online through proprioception [3, 8, 19] can also provide a more intuitive feature space; however the requirement of defining relative tradeoffs remains.

As opposed to simply reducing the complexity of defining desired behavior, the imitation learning approach seeks to learn a direct mapping to behaviors. Much previous work in this field [9, 11] has been focused on action prediction: given the current robot state and its associated perceptual features, learn to predict what ac-



(a) (b)

Fig. 2 **(a)** An example behavior (in red) overlayed on a single perceptual feature (obstacle density), and the cost function learned to reproduce the example behavior **(b)** Example behavior from time t (left) and $t+1$ (right), overlayed on a single perceptual feature (obstacle height). Future behavior is inconsistent at time t, but makes sense at time $t+1$ given additional perceptual data.

tion an expert would perform, based on examples of expert behavior. However, this approach is inherently myopic, as it assumes that all necessary information is contained in the current state. While such an approach may work for reactive planning systems, it is generally ineffective for more deliberative, goal oriented systems.

Recently, a new set of approaches have been developed for learning from demonstration based on the concept of *Inverse Optimal Control* [6]. Rather than learn a mapping from perceptual features to actions, these approaches seek to learn a mapping from perceptual features to costs, such that a planner minimizing said costs will achieve the expert demonstrated behavior (Figure 2(a)). These methods take advantage of the fact that while it is difficult for an expert to define an ordering of preferences, it is easy for an expert to demonstrate the desired behavior.

Our work makes use of the Maximum Margin Planning (MMP) [12] framework, as opposed to the Inverse Reinforcement Learning approach of [2, 1]. The primary benefits of the MMP approach are that it does not require a mixture of policies, and explicitly seeks to reproduce expert behaviors. Further, MMP has been extended to allow for nonlinear cost functions. Previous work [15] has demonstrated the applicability of the MMP framework to autonomous navigation using static perceptual data. The next section summarizes this approach, and extends it to the dynamic setting where perceptual representations evolve over time.

3 Maximum Margin Planning for Dynamic Perceptual Interpretation

This section first describes previous work in the MMP framework, and specifically the LEARCH (LEArning to seaRCH) algorithm [14], for learning a cost function to reproduce expert demonstrated behavior. Extension of LEARCH to handle dynamic, unknown environments is then presented. The MMP framework was developed in the context of Markov Decision Processes, and therefore can handle a cost function defined over state-action pairs. For the sake of simplicity and clarity, the following introduction and subsequent derivations only consider A* style planners, and costs defined over states (this is consistent with how Crusher's autonomy system oper-

ates). However, this is not a limitation of the MMP framework itself, which can be applied to any planning-based decision loop. For a full derivation, see [14].

3.1 LEARCH Algorithm

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

If an example path P_e is provided, then MMP defines the following constraint on cost functions: the cost of P_e must be lower than the cost of any other path from s_e to g_e. The structured maximum margin approach [12] encourages good generalization and prevents trivial solutions (e.g. the everywhere 0 function) by augmenting the constraint to includes a margin: 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, encoded in a loss function L_e. In this context, we define loss by how many states the two paths share. Learning a cost function then involves constrained optimization of an objective functional over C

$$\min \mathscr{O}[C] = \textsc{Reg}(C)$$

subject to the constraints

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

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

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

where REG is a regularization operator that encourages generalization in 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 necessary to enumerate these constraints. For every candidate cost function, there is a minimum cost path between two waypoints; at each step it is only necessary to enforce the constraint on this path. Further, it may not always be possible to achieve all constraints, and thus a "slack" penalty is added. Since the slack variable is tight, we may write an "unconstrained" problem that captures the constraints as penalties:

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

For linear cost functions (and convex regularization) $\mathscr{O}[C]$ is convex, and can be minimized using (sub-)gradient descent. However, linear cost functions may be in-

sufficient. Therefore, we consider the (sub-)gradient of the objective in the space of cost functions [13, 14], given by

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

where P_* is the current minimum cost plan, and δ is the dirac delta at the point of evaluation. 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 paths agree in their visitation counts at F, the functional gradient is zero.

Applying gradient descent in this space of cost functions directly would involve an extreme form of overfitting: defining a cost at every value of F encountered and involving no generalization. Instead we take a small step in a limited set of "directions" using a hypothesis space of functions mapping features to a real number. The resulting function 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 $\langle - \bigtriangledown \mathscr{O}_F[C], R_* \rangle$ between the functional gradient and elements of the space of functions under consideration. Maximizing the inner product between the functional gradient and R_* can be interpreted as a learning problem:

$$R_* = \arg\max_R \sum_{x \in P_e \cup P_*} \alpha_x y_x R(F_x) \tag{4}$$

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

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 an additional regularization to each projection. Intuitively, the regression targets are positive in places the planned path visits more than the example path, and negative in places the example path visits more. The weights on each regression target are the difference in visitation counts.

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 [14] we encourage functions that are "sparse". Thus the final cost function is of the form

$$C(F) = e^{\sum \eta_i R_i(F)} \tag{5}$$

naturally resulting in cost functions that map to $\mathbb{R}^+$. Regularization is achieved implicitly through the learning rate η_i.

With the LEARCH algorithm, multiple expert examples can be provided. Each example behavior between a single start and end waypoint defines its own objective function, and in turn its own regression targets. These targets can be tallied and a new cost function computed for each example one at a time, or all at once with a single update to the cost function.

3.2 LEARCH for Unknown, Dynamic Environments

Previous implementations of LEARCH for mobile robot navigation [15] have only considered the scenario where the mapping from states to features is static and fully known a priori. In this section, we build on [13] and extend the LEARCH algorithm to the scenario where neither of these assumptions holds, such as when features are generated from a mobile robot's perception system. The limited range inherent in onboard sensing means a great deal of the environment may be unknown; for truly complex navigation tasks, the distance between waypoints is generally at least one or two orders of magnitude larger than the sensor range. Further, changing range and point of view from environmental structures means that even once an object is within range, its perceptual features are continually changing. Finally, there are the actual dynamics of the environment: objects may move, lighting and weather conditions can change, sensors may be noisy, etc.

Since the perceptual inputs are no longer static, the robot's current plan must also be continually recomputed. The original MMP constraint must be altered in the same way: rather than enforcing the optimality of the entire example behavior once, the optimality of all example behavior must be continually enforced as the current plan is recomputed. Formally, we add a time index t to account for dynamics. F_x^t represents the perceptual features of state x at time t. P_e^t represents the example behavior starting from the current state at time t to the goal. The objective becomes

$$\min \mathscr{O}[C] = \textsc{Reg}(C) + \sum_t \left(\sum_{x \in P_e^t} C(F_x^t) - \min_{\hat{P}^t} \sum_{x \in \hat{P}^t} (C(F_x^t) - L_e^t(x)) \right) \tag{6}$$

with the extra summation over time carried into the functional gradient and its projection in Equation 4. The cost function C does not have a time index: the optimization is searching for the single cost function that best reproduces example behavior over an entire time sequence.

It is important to clarify what P_e^t represents. Until now, the terms *plan* and *behavior* have been interchangeable. This is true in the static case since the environment never evolves; as long as a plan is sufficiently followed, it does not need to be recomputed. However, in the dynamic case, an expert's plan and behavior are different notions: the plan is the currently intended future behavior, and the behavior is the result of previous plans. Therefore, P_e^t would ideally be the expert's *plan* at time t, not example behavior from time t onwards.

However, this information is generally not available: it would require the recording of an expert's current plan at each instant in time. Even if a framework for such a data collection were to be implemented, it would turn the collection of training examples into an extremely tedious and expensive process. Therefore, in practice we approximate the current plan of an expert P_e^t with the expert's behavior from t onwards. Unfortunately, this approximation can potentially create situations where the example at certain timesteps is suboptimal or inconsistent. The consequences of this inconsistency are further discussed in Section 4 (see Figure 2(b)).

Once dynamics have been accounted for, the limited range of onboard sensing can be addressed. At time t, there may be no perceptual features available corresponding to the (potentially large) section of the example path that is outside of current perceptual range. In order to perform long range navigation, a mobile robotic system must already have some approach to planning through terrain it has not directly sensed. Solutions include the use of prior knowledge [15], extrapolation from recent experience [10], or simply to assume uniform properties of unknown terrain.

Therefore, we define the set of visible states at time t as $\mathscr{V}^t$. The exact definition of visible depends on the specifics of the underlying robotic system's data fusion: $\mathscr{V}^t$ should include all states for which the cost of state x at time t is computed with the cost function currently being learned, C. For all other states $\bar{\mathscr{V}}^t$, we can assume the existence of some alternate function for computing cost, $C_{\bar{\mathscr{V}}}(x)$. The objective functional and gradient become

$$\begin{aligned}
&\min \mathscr{O}[C] = \ \textsc{Reg}(C) \ + \ \mathrm{C_E} - \mathrm{C_P} \\
&\mathrm{C_P} = \sum_t \min_{\hat{P}^t} \left(\sum_{x \in \hat{P}^t \cap \mathscr{V}^t} (C(F_x^t) - L_e^t(x)) \ + \sum_{x \in \hat{P}^t \cap \bar{\mathscr{V}}^t} C_{\bar{\mathscr{V}}}(x) \right) \\
&\mathrm{C_E} = \sum_t \left(\sum_{x \in P_e^t \cap \mathscr{V}^t} C(F_x^t) \ + \sum_{x \in P_e^t \cap \bar{\mathscr{V}}^t} C_{\bar{\mathscr{V}}}(x) \right) \qquad (7)
\end{aligned}$$

$$\bigtriangledown \mathscr{O}_F[C] = \sum_t \left(\sum_{x \in P_e \cap \mathscr{V}^t} \delta_F(F_x^t) \ - \sum_{x \in P_* \cap \mathscr{V}^t} \delta_F(F_x^t) \right) \qquad (8)$$

Since the gradient is computed with respect to C, it is only nonzero for visible states. The projection of the functional gradient onto the hypothesis space becomes

$$R_* = \arg\max_R \sum_t \sum_{x \in (P_e \cup P_*) \cap \mathscr{V}^t} \alpha_x^t y_x^t R(F_x^t) \qquad (9)$$

Although the functional gradient is zero over $\bar{\mathscr{V}}^t$, $C_{\bar{\mathscr{V}}}$ still factors into the planned behavior. Just as LEARCH learns C to recreate desired behavior when using a specific planner, it learns C to recreate behavior when using a specific $C_{\bar{\mathscr{V}}}$. However, if the example behavior is inconsistent with $C_{\bar{\mathscr{V}}}$, it will be more difficult for the planned behavior to match the example. Such an inconsistency could occur if the expert has different prior knowledge than the robot, or interprets the same knowledge differently (a problem addressed in [15]). Inconsistency can also occur due to the previously discussed mismatch between expert plans and expert behavior. Solutions to inconsistent examples are discussed in Section 4.

Contrasting the final form for R_* with that of (4) helps to summarize the changes that result in the LEARCH algorithm for dynamic environments. Specifically, a single expert demonstration from start to goal is discretized by time, with each timestep serving as an example of what behavior to plan given all data to that point in time. For each of these discretized examples, only visitation counts in visible states are used. The resulting algorithm is presented in Figure 3.

for $i = 1...N$ **do**
  foreach P_e^t **do**
    $\mathscr{M}^t = \texttt{buildCostmap}(C_{i-1}, s_e^t, g_e^t, \mathscr{F}^t)$;
    $P_*^t = \texttt{planPath}(s_e^t, g_e^t, \mathscr{M}^t)$;
    $U_-^{e,t} = \{P_e^t \bigcap \bar{P}_*^t, -1\}$, $U_+^{e,t} = \{P_*^t \bigcap \bar{P}_e^t, +1\}$;
  $R_i = \texttt{trainRegressor}(\mathscr{F}, U_+ \bigcup U_-)$;
  $C_i = C_{i-1} * e^{\eta_i R_i}$;

Fig. 3 The dynamic LEARCH algorithm

4 Imperfect and Inconsistent Demonstration

The MMP framework makes the assumption that the provided training data are examples of optimal behavior. Generally, this is not the case, as humans rarely behave in a truly optimal manner. Fundamentally, there will always be noise in human demonstration. Further, multiple examples from different environments and experts may be inconsistent with each other, due either to inconsistency in human behavior, or an incomplete perceptual description of the environment by the robot. Finally, sometimes experts are flat out wrong, and produce poor demonstration.

While MMP is not broken by poor training data, it does suffer degraded overall performance and generalization (in the same way that supervised classification performance is degraded but not broken by mislabeled training data). Attempting to have an expert sanitize the training input is disadvantageous for two reasons: it creates an additional need for human involvement, and assumes that an expert can detect all errors. Instead, this section describes modifications to the LEARCH algorithm that can increase robustness and improve generalization in the face of noisy or poor expert demonstration.

4.1 Unachievable Example Behaviors

Experts do not necessarily plan their example behavior in a manner consistent with a robot's planner: this assumption is not part of the MMP framework. However, what is assumed is that there exists some cost function that will cause the robot's planner to reproduce the behavior. This is not always the case: it is possible for an example to be *unachievable*. For example, an expert may give an inconsistently wide berth to obstacles, or make wider turns than are necessary. The result is that example paths often take slightly longer routes through similar terrain than are optimal [15].

The effect of such an unachievable example is to drive costs towards zero on the terrain in question, since this would result in any path being optimal. However, since costs are constrained to $\mathbb{R}^+$, this will never be achieved. Instead an unachievable example will have the effect of unnecessarily lowering costs over a large section of the feature space, and artificially reducing dynamic range.

This effect can be counteracted by performing a slightly different regression or classification when projecting the gradient. Instead of minimizing the weighted er-

ror, the balanced weighted error is minimized; that is, both positive and negative regression targets (corresponding to states on the planned and example path) make an equal sum contribution. Formally, R_* is replaced with R_*^B defined as

$$R_*^B = \arg\max_R \sum_t \left(\sum_{y_x^t>0} \frac{\alpha_x^t R(F_x^t)}{N_+} - \sum_{y_x^t<0} \frac{\alpha_x^t R(F_x^t)}{N_-} \right)$$
$$N_+ = \sum_t \sum_{y_x^t>0} \alpha_x^t \quad N_- = \sum_t \sum_{y_x^t<0} \alpha_x^t \tag{10}$$

This new projection will zero out the contributions of the unachievable cases described above. In the general case, the effect of balancing can be observed by rewriting the final classification in terms of the planned and example visitation counts, U_+ and U_-, and carrying the balancing through to the inputs.

$$R_* = \arg\max\langle R, U_+ - U_-\rangle \quad R_*^B = \arg\max\langle R, \frac{U_+}{N_+} - \frac{U_-}{N_-}\rangle$$
$$\langle U_+ - U_-, \frac{U_+}{N_+} - \frac{U_-}{N_-}\rangle = \frac{\langle U_+, U_+ - U_-\rangle}{N_+} + \frac{\langle U_-, U_- - U_+\rangle}{N_-}$$
$$= \frac{|U_+|^2}{N_+} + \frac{|U_-|^2}{N_-} - (\frac{1}{N_+} + \frac{1}{N_-})\langle U_+, U_-\rangle$$

The similarity between inputs to the projections is negatively correlated to the overlap of the positive and negative visitation counts, $\langle U_+, U_-\rangle$. By the Cauchy-Schwarz inequality, $\langle U_+, U_-\rangle$ is bounded by $|U_+||U_-|$, and is only tight against this bound when the visitation counts are perfectly correlated, which implies

$$\langle U_+, U_-\rangle = |U_+||U_-| \iff U_- = \pm\kappa U_+ \implies |U_-| = \kappa|U_+| \text{ , } N_- = \kappa N_+$$

for some scalar κ. By substitution

$$\frac{|U_+|^2}{N_+} + \frac{|U_-|^2}{N_-} - (\frac{1}{N_+} + \frac{1}{N_-})\langle U_+, U_-\rangle \geq \frac{|U_+|^2}{N_+} + \frac{|U_-|^2}{N_-} - (\frac{1}{N_+} + \frac{1}{N_-})|U_+||U_-|$$
$$= \frac{|U_+|^2}{N_+} + \frac{\kappa^2|U_+|^2}{\kappa N_+} - (\frac{1}{N_+} + \frac{1}{\kappa N_+})\kappa|U_+||U_+|$$
$$= \frac{|U_+|^2}{N_+} + \frac{\kappa|U_+|^2}{N_+} - \frac{|U_+|^2}{N_+} - \frac{\kappa|U_+|^2}{N_+} = 0$$

therefore

$$\langle U_+ - U_-, \frac{U_+}{N_+} - \frac{U_-}{N_-}\rangle \geq 0$$

When there exists clear differentiation between what features should have their costs increased and decreased, the projection inputs will be similar. As the example and current planned behaviors travel over increasingly similar terrain, the inputs diverge; the contribution of the balanced projection to the current cost function will level out,

while that of the unbalanced projection will increase in the direction of the longer path. This effect is observed empirically in Section 5.

4.2 Replanning with Corridor Constraints

A balanced projection can account for large scale suboptimality in demonstration. However, suboptimality can also occur at a smaller scale. It is unreasonable to ever expect a human to drive the exact perfect path; it is often the case that a plan that travels through neighboring or nearby states would be a slightly better example. What is needed is an approach that smoothes out small scale noise in expert demonstration, producing a better training example. Such a smoothed example can be derived from expert demonstration by redefining what the example represents: instead of example behavior being interpreted as the exact optimal behavior, it can be interpreted as a behavior that is close to optimal. The exact definition of close depends on the state space; the loss function will always provide at least one possible metric.

For example, if the state space is $\mathbb{R}^n$, then Euclidean distance is a natural metric. Rather than an example defining the exact optimal path, it would define a corridor in which the optimal path exists. A new desired behavior can be derived from the example at each learning iteration by choosing the current optimal behavior that is sufficiently close. Formally, $\mathrm{C_E}$ in (7) is redefined as

$$\begin{aligned} \mathrm{C_E} &= \sum_t \min_{\hat{P}_e^t \in \mathscr{N}_e^t} \left(\sum_{x \in \hat{P}_e^t \cap \mathscr{V}^t} C(F_x^t) + \sum_{x \in \hat{P}_e^t \cap \bar{\mathscr{V}}^t} C_{\bar{\mathscr{V}}}(x) \right) \\ \mathscr{N}_e^t &= \{P \mid \; ||(P \bigcap \mathscr{V}^t) - (P_e^t \bigcap \mathscr{V}^t)|| < \beta\} \end{aligned} \tag{11}$$

where β is a threshold on the metric over paths. The result of this modification is that the example path is replanned at each iteration of the LEARCH algorithm; however, only paths within $\mathscr{N}_e^t$ are considered during replanning.

It should be noted that $\mathscr{N}_e^t$ only defines closeness over $\mathscr{V}^t$. Behavior outside of $\mathscr{V}^t$ does not directly affect the gradient, but does affect the difference in cost between the current (replanned) example and planned behavior. Therefore, by performing a replanning step (even with $\beta = 0$), example behavior can be made consistent with $C_{\bar{\mathscr{V}}}$ without compromising its effectiveness as an example within $\mathscr{V}^t$.

4.3 Filtering for Inconsistent Examples

By always performing a replanning step, it can be ensured that any remaining inconsistency is due to either a mismatch between an expert's planned and actual behavior at time t, a disagreement between the expert's and the robot's underlying perception of the world, or expert error. Figure 2(b) provides a simple example: at time t, the expert likely planned to drive straight, but was forced to replan at time $t+1$ as new obstacles were observed. The assumption that the expert behavior from time t on-

ward matches the plan is false, and results in behavior at time t being inconsistent with the behavior exhibited at other timesteps. While tiny changes in an expert's plan can be corrected by replanning the example, there is no way to correct for a drastic case such as a cul de sac.

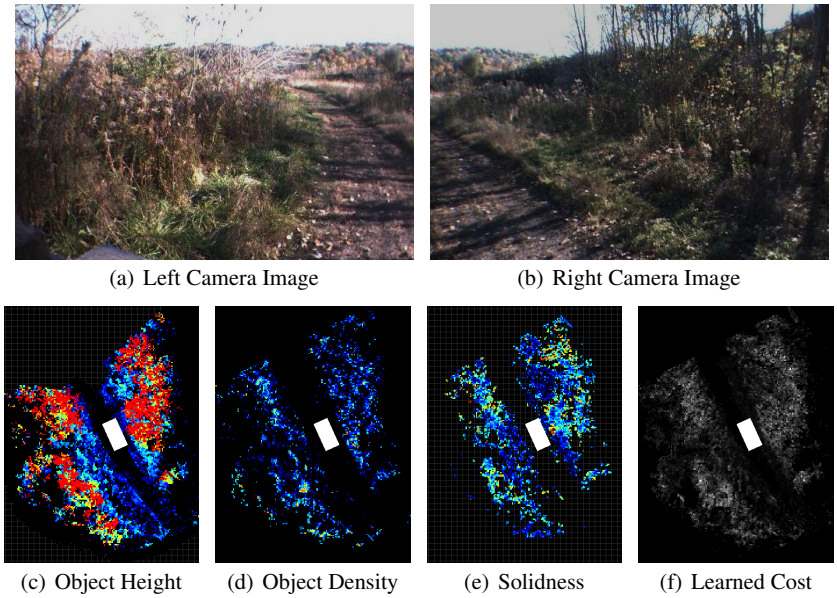
However, the inconsistency of such timesteps provides a basis for their filtering and removal. Specifically, it can be assumed that a human expert will plan in a fairly consistent manner during a single example traverse. If the behavior from a single timestep or small set of timesteps is inconsistent with the demonstrated behavior at other timesteps, then these inconsistent timesteps can be filtered. Inconsistency can be quantitatively defined by observing each timestep's contribution to the objective functional (its slack penalty). If the penalty at a single timestep is a statistical outlier from other timesteps, then that timestep can be seen as inconsistent with the constraints implied by the rest of the example behavior.

Therefore, the following filtering heuristic is proposed. First, attempt to learn a cost function over all timesteps of a single example behavior, using a more complex hypothesis space than intended for the final cost function, and identify timesteps whose penalties are statistical outliers. As these timesteps are inconsistent within an overly complex hypothesis space, there is evidence that the inconsistency is in the example and not for lack of expressiveness in the cost function. Therefore, these timesteps can be removed. This process can be repeated for each example behavior, and only remaining timesteps used in the final training. Experimental results of this filtering approach are presented in the next section.

5 Experimental Results

Our imitation learning approach was implemented for the Crusher autonomous platform (Figure 1) [18]. Crusher's base perception system consists of 6 LiDAR scanners and 4 sets of cameras which are processed to provide a discretized set of geometric and semantic features at a range of up to 20 m (Figure 4). Training data in the form of expert example behaviors was gathered by having Crusher's safety operator teleoperate the vehicle through sets of waypoints. Different training examples were collected over a period of months in varying locations and weather conditions, and with 3 different operators at one time or another. During data collection, all raw sensor data was logged along with the example path. Perceptual features were then produced offline by feeding the raw sensor data through Crusher's perception software. In this way, the base perception system and its features can be modified and improved without having to recollect new training data; the raw data is just reprocessed, and a cost function learned for the new features. Figure 4 provides a simple example of both perceptual features and the resulting (learned) cost.

This set of examples was used to perform a series of offline experiments to validate additions to the MMP framework. The average loss over a path (as defined in Equation 1) was used to evaluate the performance of different variations of the LEARCH algorithm on a large validation set of example behaviors. These results are presented in Figure 5. Fig. 5(a) demonstrates the effectiveness of performing



(a) Left Camera Image (b) Right Camera Image

(c) Object Height (d) Object Density (e) Solidness (f) Learned Cost

Fig. 4 An example of learned perception cost. A simple scene depicted in (a),(b) results in perceptual features (a small subset shown in (c) - (e)) that are mapped into cost (h).

balanced as opposed to unbalanced regression, as the balanced version has superior validation performance. Fig 5(b) demonstrates the benefit of replanning with corridor constraints. With a small corridor size, the algorithm is able to smooth out some of the noise in demonstrated behavior, and improve generalization. As the corridor size increases, the algorithm begins to oversmooth, resulting in decreasing validation performance. In this way, validation data can be used to automatically determine the optimal corridor size. An experiment was also performed to assess the effectiveness of filtering. A single expert behavior was used to learn a cost function, first with no filtering, and then with approximately 10% of its timesteps automatically filtered. Without filtering, the training loss (computed only over unfiltered timesteps) and validation loss were 0.532 and 0.596 respectively. With filtering, these numbers improved to 0.493 and 0.582, indicating further enhanced generalization.

As cost evaluations must be performed online in real time, the computational cost of an evaluation is an important consideration. As a sum of linear functions is another linear function, a linear hypothesis space is computationally advantageous. However, this negates one of the primary benefits of LEARCH (nonlinear cost functions). Therefore, additional experiments were performed with the approach used in [15]: linear regression was used to increase efficiency and generalization, but a feature learning phase was added. Occasional projections of the functional gradient are performed with simple regression trees; these regression trees are then used as a new feature instead of being added directly to the cost function. In this way, future learning iterations can control the contribution of each regression tree feature, and the computational cost of an evaluation is minimized. This approach is similar to the

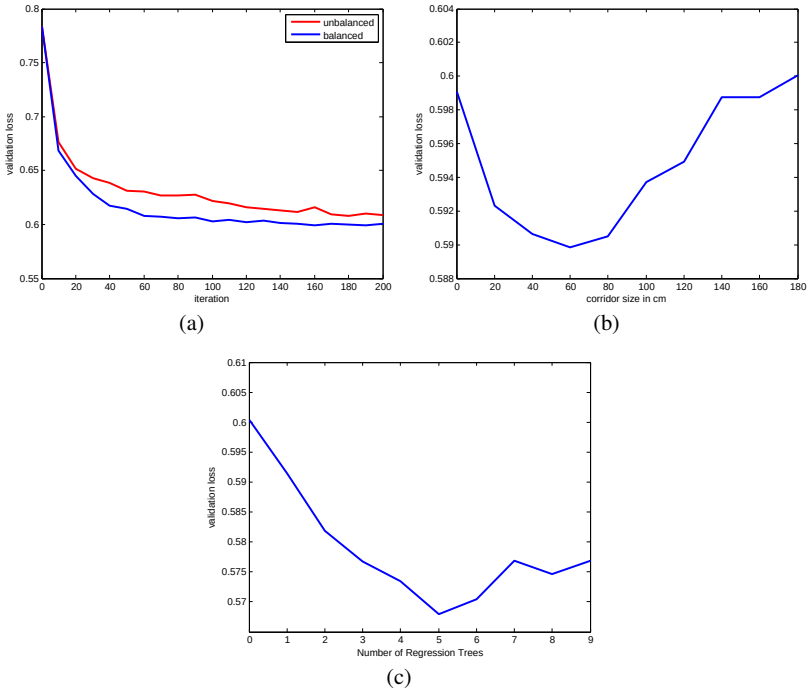


Fig. 5 Results of offline experiments. **(a)** Validation Loss during learning for the balanced and unbalanced functional gradient projection **(b)** Validation Loss as a function of the replanning corridor size **(c)** Validation Loss as a function of the number of regression trees.

way cost functions are often engineered, with specific rules added to handle special cases. Figure 5(c) shows validation loss as a function of the number of added regression tree features. At first, additional features improve the validation performance; eventually too many features results in overfitting.

The training set was also used to learn a cost function to run onboard Crusher, which uses a hybrid global and local planning system similar to [7]. Details of the implementation of LEARCH with this planning system can be found in [16]. Originally, the function mapping perceptual features to costs onboard Crusher was hand engineered. This cost function was developed and continually retuned by multiple students, staff, and faculty; this process required hundreds of man-hours over a period of more than 3 years to achieve and maintain a high level of autonomous performance [18]. Much of this maintenance involved retuning the cost function in newly encountered regions of the perceptual feature space observed when first testing in novel terrain (while still trying to maintain good performance in previously encountered terrain). In contrast, the entirety of the training set used for imitation learning involved less than 1 hour of expert driving examples, over a total distance of less than 3 km. This seemingly small amount of human demonstration is sufficient due to the numerous constraints implied by each example behavior: a few kilometers of demonstration provides hundreds of thousands of examples of states to traverse, and millions more examples of states that were avoided. Additionally, if testing in a

Statistic for Comparison	Engineered	Learned	P-value
Avg. Distance Per Waypoint(m)	130.7	124.3	0.07
Avg. Cmd. Vel.($\frac{m}{s}$),Ang. Vel.($\frac{\circ}{s}$)	3.24,6.55	3.39,5.08	0.15,0.0
Avg. Lateral Vel./Accel.($\frac{m}{s^2}$)	0.181/1.34	0.17/1.31	0.0/0.0
Avg. Roll/Pitch($\circ$)	4.05/2.21	3.90/2.18	0.99/0.57
Avg. Slip Ratio	1.131	1.129	0.81
Interventions (Per Waypoint)	8 (0.027)	10 (0.034)	0.48

Table 1 Per-waypoint averages of various metrics of autonomous performance, demonstrating the difference between operating with an engineered and a learned cost function. The difference in metrics related to efficiency of autonomous traverse are statistically significant, while those related to autonomous vehicle safety are not. Statistics for additional metrics are provided in [16].

novel environment demonstrates the existing training set to be insufficient, adapting to the new terrain requires only the collection of a few additional minutes of expert demonstration, and then no further human involvement.

The effectiveness of learned cost functions was validated during hundreds of kilometers of autonomous traverse through highly varying terrains across the continental U.S. During these trials, Crusher used learned cost functions to interpret both onboard and prior overhead perceptual data [15], and reliably navigate between widely spaced waypoints. Additionally, a large set of direct comparison trials were performed to evaluate the online performance of a learned cost function in contrast to the engineered one. During nearly 40 km of autonomous navigation with each cost function, a variety of statistics describing each run were produced (unlike many other comparisons of mobile robot performance, cost itself cannot be used). Table 1 lists some of these statistics and the significance of the difference between systems (treating each waypoint to waypoint traverse as an independent trial). More detailed statistics from these experiments can be found in [16].

The biggest difference in performance when using the two cost functions was with respect to turning: the engineered system was more apt to turn harder to avoid perceived hazards, resulting in a slower traverse and more lateral motion. The effect of this difference on the number of safety interventions and other proprioceptive measures of incurred hazards was not statistically significant (implying that the engineered function may produced more false positives). Therefore, we claim a similar high level of autonomous performance between the two costing approaches, but with orders of magnitude less human involvement in the learned system.

6 Conclusion

This paper addresses the task of interpreting perceptual data for use in autonomous navigation. We have presented a dynamic imitation learning approach that achieves equivalent performance to human engineering with far less manual interaction. The approach easily adapts to new perceptual features, without the need for additional human involvement. Future work will explore the application of this approach to learning cost functions over full state-action pairs. The behavior of an autonomous navigation system is defined not only by which terrain it prefers, but by which motions it prefers (e.g. minimum curvature); by learning all preferences at once from

human demonstration, we hope to further improve the robustness of autonomous navigation. Additionally, the use of active learning to solicit expert demonstration will be investigated. Finally, the combination of our approach with self-supervised learning will be explored, to create mobile systems which are trained once and improve with experience.

This work was sponsored by 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. Abbeel, P., Dolgov, D., Ng, A.Y., Thrun, S.: Apprenticeship learning for motion planning with application to parking lot navigation. In: Conference on Intelligent Robots and Systems (2008)
2. Abbeel, P., Ng, A.Y.: Apprenticeship learning via inverse reinforcement learning. In: International Conference on Machine learning (2004)
3. Brooks, C., Iagnemma, K.: Self-supervised classification for planetary rover terrain sensing. In: IEEE Aerospace Conference (2007)
4. Goldberg, S., Maimone, M., Matthies, L.: Stereo vision and rover navigation software for planetary exploration. Aerospace Conference Proceedings, 2002. IEEE **5**, 2025–2036 (2002)
5. Howard, A., Seraji, H.: Vision-based terrain characterization and traversability assessment. Journal of Robotic Systems **18** (2001)
6. Kalman, R.: When is a linear control system optimal? Trans. ASME, J. Basic Engrg. **86**, 51–60 (1964)
7. Kelly, A., Amidi, O., Happold, M., Herman, H., Pilarski, T., Rander, P., Stentz, A., Vallidis, N., Warner, R.: Toward reliable autonomous vehicles operating in challenging environments. In: International Symposium on Experimental Robotics (ISER). Singapore (2004)
8. Kim, D., Sun, J., Oh, S.M., Rehg, J.M., Bobick, A.F.: Traversability classification using unsupervised on-line visual learning. In: IEEE Conference on Robotics and Automation (2006)
9. LeCun, Y., Muller, U., Ben, J., Cosatto, E., Flepp, B.: Off-road obstacle avoidance through end-to-end learning. In: Advances in Neural Information Processing Systems 18 (2006)
10. Nabbe, B., Kumar, S., Hebert, M.: Path planning with hallucinated worlds. In: Proceedings: IEEE/RSJ International Conference on Intelligent Robots and Systems (2004)
11. Pomerleau, D.: Alvinn: An autonomous land vehicle in a neural network. In: Advances in Neural Information Processing Systems 1 (1989)
12. Ratliff, N., Bagnell, J., Zinkevich, M.: Maximum margin planning. In: International Conference on Machine Learning (2006)
13. Ratliff, N., Bradley, D., Bagnell, J., Chestnutt, J.: Boosting structured prediction for imitation learning. In: Advances in Neural Information Processing Systems 19. Cambridge, MA (2007)
14. Ratliff, N.D., Silver, D., Bagnell, J.A.: Learning to search: Functional gradient techniques for imitation learning. Autonomous Robots **27**(1), 25–53 (2009)
15. Silver, D., Bagnell, J.A., Stentz, A.: High performance outdoor navigation from overhead data using imitation learning. In: Proceedings of Robotics Science and Systems (2008)
16. Silver, D., Bagnell, J.A., Stentz, A.: Applied imitation learning for autonomous navigation in complex natural terrain. In: Field and Service Robotics (2009)
17. Singh, S., Simmons, R., Smith, T., Stentz, A., Verma, V., Yahja, A., Schwehr, K.: Recent progress in local and global traversability for planetary rovers. IEEE Conference on Robotics and Automation (2000)
18. Stentz, A., Bares, J., Pilarski, T., Stager, D.: The crusher system for autonomous navigation. In: AUVSIs Unmanned Systems (2007)
19. Wellington, C., Stentz, A.: Online adaptive rough-terrain navigation vegetation. In: Proceedings of the IEEE International Conference on Robotics and Automation (2004)