Proceedings of Machine Learning Research vol 120:1-11, 2020 2nd Annual Conference on Learning for Dynamics and Control

Learning Navigation Costs from Demonstration with Semantic Observations

Tianyu Wang TIW161@ENG.UCSD.EDU
Vikas Dhiman VDHIMAN@ENG.UCSD.EDU
Nikolay Atanasov NATANASOV@ENG.UCSD.EDU
Department of Electrical and Computer Engineering, University of California San Diego, La Jolla, CA 92093

Editors: A. Bayen, A. Jadbabaie, G. J. Pappas, P. Parrilo, B. Recht, C. Tomlin, M. Zeilinger

Abstract

This paper focuses on inverse reinforcement learning (IRL) for autonomous robot navigation using semantic observations. The objective is to infer a cost function that explains demonstrated behavior while relying only on the expert's observations and state-control trajectory. We develop a map encoder, which infers semantic class probabilities from the observation sequence, and a cost encoder, defined as deep neural network over the semantic features. Since the expert cost is not directly observable, the representation parameters can only be optimized by differentiating the error between demonstrated controls and a control policy computed from the cost estimate. The error is optimized using a closed-form subgradient computed only over a subset of promising states via a motion planning algorithm. We show that our approach learns to follow traffic rules in the autonomous driving CARLA simulator by relying on semantic observations of cars, sidewalks and road lanes.

Keywords: Inverse reinforcement learning, semantic mapping, learning from demonstration

1. Introduction

Autonomous systems operating in unstructured, partially observed, and changing real-world environments need an understanding of context to evaluate the safety, utility, and efficiency of their performance. For example, while a bipedal robot may navigate along sidewalks, an autonomous car needs to follow the road lane structure and the traffic signs. Designing a cost function that encodes such rules by hand is cumbersome, if not infeasible, especially for complex tasks. However, it is often possible to obtain demonstrations of desirable behavior that indirectly capture the role of semantic context in the task execution. Semantic labels provide rich information about the relationship between object entities and their surroundings. In this work, we consider an inverse reinforcement learning (IRL) problem in which observations containing semantic information about the environment are available.

There has been significant progress in semantic segmentation techniques, including deep neural networks for RGB image segmentation (Papandreou et al., 2015; Badrinarayanan et al., 2017; Chen et al., 2018) or point cloud labeling via a 2D spherical depth projection (Wu et al., 2018; Dohan et al., 2015). Maps that store semantic information can be generated from segmented images (Sengupta et al., 2012; Lu et al., 2019). Gan et al. (2019); Sun et al. (2018) generalize binary occupancy grid mapping (Hornung et al., 2013) to multi-class semantic mapping in 3D. In this work, we parameterize the navigation cost of an autonomous vehicle as a nonlinear function of such semantic features to explain the demonstrations of an expert.

© 2020 T. Wang, V. Dhiman & N. Atanasov.

Learning a cost function from demonstration requires a control policy that is differentiable with respect to the cost parameters. Computing policy derivatives has been addressed by several sucessful IRL approaches (Neu and Szepesvári, 2007; Ratliff et al., 2006; Ziebart et al., 2008). Early works assume that the cost is linear in the feature vector and aim at matching the feature expectations of the learned and expert policies. Ratliff et al. (2006) computes subgradients of planning algorithms so that expected reward of an expert policy is better than any other policy by a margin. Value iteration networks (VIN) (Tamar et al., 2016) show that the value iteration algorithm can be approximated by a series of convolution and maxpooling layers, allowing automatic differentiation to learn the cost function end-to-end. Ziebart et al. (2008) develops a dynamic programming algorithm to maximize the likelihood of observed expert data and learns a policy of maximum entropy (MaxEnt) distribution. Many works (Levine et al., 2011; Wulfmeier et al., 2016; Song, 2019) extend MaxEnt to learn a nonlinear cost using Gaussian Processes or deep neural networks. Finn et al. (2016) uses sample-based approximation of the MaxEnt objective on high-dimensional continuous systems. However, the cost in most existing work is learned offline using full observation sequences from the expert demonstrations. A *major contribution* of our work is to develop cost representations and planning algorithms that rely only on causal partial observations.

Achieving safe and robust navigation is directly coupled with the quality of the environment representation and the cost function specifying desirable behaviors. The traditional approach combines geometric mapping of occupancy probability or distance to the nearest obstacle (Hornung et al., 2013; Oleynikova et al., 2017) with hand-specified planning cost functions. Recent advances in deep reinforcement learning demonstrated that control inputs may be predicted directly from sensory observations (Levine et al., 2016). However, special model designs (Khan et al., 2018) that serve as a latent map are needed in navigation tasks where simple reactive policies are not feasible. Gupta et al. (2017) decompose visual navigation into two separate stages explicitly: mapping the environment from first-person RGB images and planning through the constructed map with VIN. Our model also seperates the two stages but integrates semantic information to obtain a richer map representation. In addition, Wang et al. (2020) propose a differentiable mapping and planning framework to learn the expert cost function. They parameterize cost function as a neural network over the binary occupancy map probability, which is integrated from previous partial observations. They further propose an efficient A* (Hart et al., 1968) planning algorithm that computes the policy at the current state and backpropagates gradients in closed-form to optimize the cost parameters. We extend their work by incorporating semantic observation to the map representation and evaluating the model in the CARLA autonomous driving simulator (Dosovitskiy et al., 2017).

We propose a model that learns to navigate from first-person semantic observations and make the following contributions. First, we propose a cost function representation composed of a *map encoder*, capturing semantic class probabilities from the streaming observations, and a *cost encoder*, defined as a deep neural network over the semantic features. Second, we optimize the cost parameters using a closed-form subgradient of the cost-to-go only over a subset of promising states, obtained by an efficient planning algorithm. Finally, we verify our model in autonomous navigation experiments in urban environments provided by the CARLA simulator (Dosovitskiy et al., 2017).

2. Problem Formulation

Consider a robot navigating in an unknown environment $\mathcal{X}$ with the task of reaching a goal state $\boldsymbol{x}_g \in \mathcal{X}$. Let $\boldsymbol{x}_t \in \mathcal{X}$ be the robot state, capturing its pose, twist, etc., at discrete time t. For a

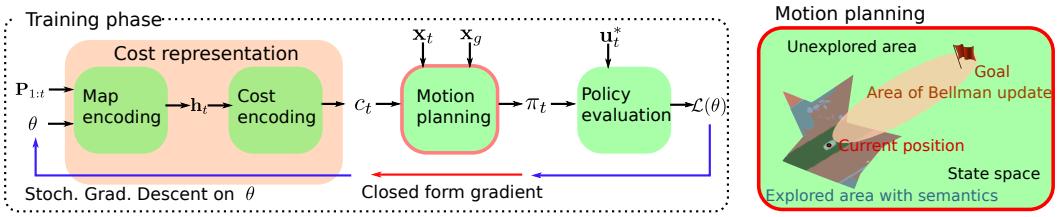

Figure 1: Architecture for cost function learning from demonstrations with semantic observations. Our main contribution is a cost representation, combining a probabilistic *semantic map encoder*, with recurrent dependence on semantic observations $\boldsymbol{P}_{1:t}$, and a *cost encoder*, defined over the semantic features. Efficient forward policy computation and closed-form subgradient backpropagation are used to optimize the cost representation in order to explain expert behavior.

given control input $\boldsymbol{u}_t \in \mathcal{U}$ where $\mathcal{U}$ is assumed finite, the robot state evolves according to known deterministic dynamics: $\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t)$. Let $\mathcal{K} = \{0, 1, 2, \ldots, K\}$ be a set of class labels, where $k = 0$ denotes "free" space and $k \geq 1$ denotes a particular semantic class such as car or tree. Let $m^* : \mathcal{X} \to \mathcal{K}$ be a function specifying the *true* semantic occupancy of the environment by labeling states with semantic classes and $\mathcal{M}$ be the space of possible environment realizations m^*. Let $c^* : \mathcal{X} \times \mathcal{U} \times \mathcal{M} \to \mathbb{R}_{\geq 0}$ be a cost function specifying desirable robot behavior in a given environment, e.g., according to an expert user or an optimal design. We assume that the robot does not have access to either the true semantic map m^* or the true cost function c^*. However, the robot is able to obtain point cloud observations $\boldsymbol{P}_t = \{(\boldsymbol{p}_l, \boldsymbol{y}_l)\}_l \in \mathcal{P}$ at each step t, where $\boldsymbol{p}_l \in \mathcal{X}$ is the measurement location and $\boldsymbol{y}_l$ is an observed semantic likelihood such that $\boldsymbol{y}_l = \left[y_l^1, \ldots, y_l^K\right]^T, y_l^k \geq 0, \sum_{k=1}^K y_l^k = 1$, whose support is $\mathcal{K} \setminus \{0\}$. In practice, $\boldsymbol{y}_l$ can be obtained from a semantic segmentation algorithm (Papandreou et al., 2015; Badrinarayanan et al., 2017; Chen et al., 2018) that predicts the semantic class of the corresponding measurement location $\boldsymbol{p}_l$. The observed point cloud $\boldsymbol{P}_t$ depends on the robot state $\boldsymbol{x}_t$ and the environment realization m^*. Given a training set $\mathcal{D} := \left\{(\boldsymbol{x}_{t,n}, \boldsymbol{u}_{t,n}^*, \boldsymbol{P}_{t,n}, \boldsymbol{x}_{g,n})\right\}_{t=1,n=1}^{T_n,N}$ of N expert trajectories with length T_n to demonstrate desirable behavior, our goal is to

- learn a cost function estimate $c_t : \mathcal{X} \times \mathcal{U} \times \mathcal{P}^t \times \Theta \to \mathbb{R}_{\geq 0}$ that depends on an observation sequence $\boldsymbol{P}_{1:t}$ from the true latent environment and is parameterized by $\boldsymbol{\theta} \in \Theta$,
- design a stochastic policy π_t from c_t such that the robot behavior under π_t matches the prior experience $\mathcal{D}$.

To balance exploration in partially observable environments with exploitation of promising controls, we specify π_t as a Boltzmann policy (Ramachandran and Amir, 2007; Neu and Szepesvári, 2007) associated with the cost c_t, $\pi_t(\boldsymbol{u}_t|\boldsymbol{x}_t; \boldsymbol{P}_{1:t}, \boldsymbol{\theta}) = \frac{\exp(-Q_t^*(\boldsymbol{x}_t, \boldsymbol{u}_t; \boldsymbol{P}_{1:t}, \boldsymbol{\theta}))}{\sum_{\boldsymbol{u} \in \mathcal{U}} \exp(-Q_t^*(\boldsymbol{x}_t, \boldsymbol{u}; \boldsymbol{P}_{1:t}, \boldsymbol{\theta}))}$, where the optimal cost-to-go function Q_t^* is:

$$Q_t^*(\boldsymbol{x}_t, \boldsymbol{u}_t; \boldsymbol{P}_{1:t}, \boldsymbol{\theta}) := \min_{\boldsymbol{u}_{t+1:T-1}} \sum_{k=t}^{T-1} c_t(\boldsymbol{x}_k, \boldsymbol{u}_k; \boldsymbol{P}_{1:t}, \boldsymbol{\theta}) \;\text{ s.t. } \boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k),\; \boldsymbol{x}_T = \boldsymbol{x}_g. \quad (1)$$

Problem *Given demonstrations $\mathcal{D}$, optimize the cost function parameters $\boldsymbol{\theta}$ so that log-likelihood of the demonstrated controls $\boldsymbol{u}_{t,n}^*$ is maximized under the robot policies $\pi_{t,n}$:*

$$\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) := -\sum_{n=1}^{N} \sum_{t=1}^{T_n} \log \pi_{t,n}(\boldsymbol{u}_{t,n}^*|\boldsymbol{x}_{t,n}; \boldsymbol{P}_{1:t,n}, \boldsymbol{\theta}). \quad (2)$$

The problem setup is illustrated in Fig. 1. While Eqn. (1) is a standard deterministic shortest path (DSP) problem, the challenge is to make it differentiable with respect to $\boldsymbol{\theta}$, which is necessary for the loss in (2) to propagate back through the DSP problem to update the cost parameters $\boldsymbol{\theta}$. Once the parameters are optimized, the robot can generalize to navigation tasks in new partially observable environments by evaluating the cost c_t based on the observations $\boldsymbol{P}_{1:t}$ iteratively and (re)computing the associated policy π_t.

3. Cost Function Representation

We propose a cost function representation with two components: a *semantic map encoder* with parameters $\boldsymbol{\psi}$ and a *cost encoder* with parameters $\boldsymbol{\phi}$. The model is differentiable by design, allowing its parameters to be optimized by the subsequent planning algorithm described in Sec. 4.

3.1. Semantic Map Encoder

The semantic probability of different environment areas is encoded in a hidden state $\boldsymbol{h}_t$ given the trajectory and observations $\boldsymbol{x}_{1:t}, \boldsymbol{P}_{1:t}$. Specifically, we discretize the state space $\mathcal{X}$ into J cells and let $\boldsymbol{m} = \left[m^1, \ldots, m^J\right]^T \in \mathcal{K}^J$ be the random vector of true semantic labels over the cells. Since $\boldsymbol{m}$ is unknown to the robot, we maintain the semantic occupancy posterior $\mathbb{P}(\boldsymbol{m} = \boldsymbol{k}|\boldsymbol{x}_{1:t}, \boldsymbol{P}_{1:t})$ where $\boldsymbol{k} = \left[k^1, \ldots, k^J\right]^T, k^j \in \mathcal{K}$, given the history of states $\boldsymbol{x}_{1:t}$ and observations $\boldsymbol{P}_{1:t}$. The representation complexity may be simplified significantly if one assumes independence among the map cells m^j: $\mathbb{P}(\boldsymbol{m} = \boldsymbol{k}|\boldsymbol{x}_{1:t}, \boldsymbol{P}_{1:t}) = \prod_{j=1}^{J} \mathbb{P}(m^j = k^j|\boldsymbol{x}_{1:t}, \boldsymbol{P}_{1:t})$.

Inspired by the binary occupancy grid mapping (Thrun et al., 2005; Hornung et al., 2013), we extend the recurrent updates for the multi-class semantic probability of each cell m^j.

Definition 1 *The log-odds ratio of semantic classes associated with cell m^j at time t is*

$$\boldsymbol{h}_{t,j} = \left[h_{t,j}^0, \ldots, h_{t,j}^K\right]^T, \quad h_{t,j}^k := \log \frac{\mathbb{P}(m^j = k|\boldsymbol{x}_{1:t}, \boldsymbol{P}_{1:t})}{\mathbb{P}(m^j = 0|\boldsymbol{x}_{1:t}, \boldsymbol{P}_{1:t})} \text{ for } k \in \mathcal{K}. \tag{3}$$

Its recurrent Bayesian update is $h_{t+1,j}^k = h_{t,j}^k + \log \frac{p(\boldsymbol{P}_{t+1}|m^j=k, \boldsymbol{x}_{t+1})}{p(\boldsymbol{P}_{t+1}|m^j=0, \boldsymbol{x}_{t+1})}$. Note that by definition $h_{t,j}^0 = 0$. The update increment is a log-odds observation model and we assume the observation $\boldsymbol{P}_{t+1}$ given the cell m^j is independent of the previous observations $\boldsymbol{P}_{1:t}$. The semantic class posterior can be recovered from the semantic log-odds ratio $\boldsymbol{h}_{t,j}$ via a softmax function $\mathbb{P}(m^j = k|\boldsymbol{x}_{1:t}, \boldsymbol{P}_{1:t}) = \sigma^k(\boldsymbol{h}_{t,j})$, where $\sigma : \mathbb{R}^{K+1} \to \mathbb{R}^{K+1}$ satisfies the following properties

$$\sigma(\boldsymbol{z}) = \left[\sigma^0(\boldsymbol{z}), \ldots, \sigma^K(\boldsymbol{z})\right]^T, \ \sigma^k(\boldsymbol{z}) = \frac{\exp(z^k)}{\sum_{k' \in \mathcal{K}} \exp(z^{k'})}, \ \log \frac{\sigma^k(\boldsymbol{z})}{\sigma^{k'}(\boldsymbol{z})} = z^k - z^{k'}. \tag{4}$$

We provide a simple observation model to instantiate Eq. (3). Consider all cells m^j that lie on the ray between robot state $\boldsymbol{x}$ and a labeled point $(\boldsymbol{p}_l, \boldsymbol{y}_l)$ in the point cloud $\boldsymbol{P}$. Let $d(\boldsymbol{x}, m^j)$ be the distance between the robot position and the center of mass of the cell m^j.

Definition 2 *The inverse observation model relating the label of cell m^j to the ray between robot state $\boldsymbol{x}$ and labeled point $(\boldsymbol{p}_l, \boldsymbol{y}_l)$ is defined as a softmax function with parameters $\boldsymbol{\psi}_l$, scaled by the distance, $\delta p_l = d(m^j, \boldsymbol{x}) - \|\boldsymbol{p}_l\|_2$, which is truncated at a threshold ϵ:*

$$\mathbb{P}(m^j = k|\boldsymbol{x}, (\boldsymbol{p}_l, \boldsymbol{y}_l)) = \begin{cases} \sigma^k(diag(\boldsymbol{\psi}_l)\boldsymbol{y}_l \delta p_l) & \text{if } \delta p_l \leq \epsilon \\ \sigma^k(\boldsymbol{h}_{0,j}) & \text{if } \delta p_l > \epsilon \end{cases}. \tag{5}$$

The function $diag(\cdot)$ returns a diagonal matrix from a vector and the prior occupancy log-odds ratio $\boldsymbol{h}_{0,j}$ depends on the environment (e.g., $\boldsymbol{h}_{0,j} = \mathbf{0}$ specifies a uniform prior over the semantic classes).

Proposition 3 *Given the definitions of the log-odds ratio in Eq. (3) and the inverse observation model in Eq. (5), the log-odds update rule for the semantic probability at cell m^j is $\boldsymbol{h}_{t+1,j} = \boldsymbol{h}_{t,j} + \sum_{(\boldsymbol{p}_l,\boldsymbol{y}_l)\in\boldsymbol{P}_{t+1}} \left[\boldsymbol{g}_j(\boldsymbol{x}_t,(\boldsymbol{p}_l,\boldsymbol{y}_l)) - \boldsymbol{h}_{0,j}\right]$, where the log-odds inverse observation model for cells m^j along the ray from $\boldsymbol{x}_t$ to $\boldsymbol{p}_l$ can be simplied using (4) as:*

$$\boldsymbol{g}_j(\boldsymbol{x}_t,(\boldsymbol{p}_l,\boldsymbol{y}_l)) = \begin{cases} diag(\boldsymbol{\psi}_l)\boldsymbol{y}_l\delta p_l & \text{if } \delta p_l \leq \epsilon \\ \boldsymbol{h}_{0,j} & \text{if } \delta p_l > \epsilon \end{cases}. \tag{6}$$

A more expressive multi-layer neural network may be used to parameterize the inverse observation model instead of the linear transformation $diag(\boldsymbol{\psi}_l)\boldsymbol{y}_l\delta p_l$ of the semantic probability and distance differential along the l-th ray in Eq (5):

$$\boldsymbol{g}_j(\boldsymbol{x}_t,(\boldsymbol{p}_l,\boldsymbol{y}_l);\boldsymbol{\psi}_l) = \begin{cases} \mathbf{NN}(\boldsymbol{y}_l,\boldsymbol{p}_l,d(\boldsymbol{x}_t,m^j);\boldsymbol{\psi}_l) & \text{if } \delta p_l \leq \epsilon \\ \boldsymbol{h}_{0,j} & \text{if } \delta p_l > \epsilon \end{cases}. \tag{7}$$

In summary, the map encoder starts with prior log-odds $\boldsymbol{h}_0$, updates them recurrently via $\boldsymbol{h}_{t+1} = \boldsymbol{h}_t + \boldsymbol{g}(\boldsymbol{x}_t,\boldsymbol{P}_t;\boldsymbol{\psi}) - \boldsymbol{h}_0$, where the inverse sensor log-odds $\boldsymbol{g}_j(\boldsymbol{x}_t,(\boldsymbol{p}_l,\boldsymbol{y}_l);\boldsymbol{\psi}_l)$ is specified for the j-th cell along the l-th ray in (6) or (7). The posterior $\mathbb{P}(\boldsymbol{m} = \boldsymbol{k}|\boldsymbol{x}_{1:t},\boldsymbol{P}_{1:t})$ is the softmax of $\boldsymbol{h}_t$.

3.2. Cost Encoder

The cost encoder uses the semantic occupancy grid prosterior $\sigma(\boldsymbol{h}_t)$ to define the cost function estimate $c_t(\boldsymbol{x},\boldsymbol{u})$ at a given state-control pair $(\boldsymbol{x},\boldsymbol{u})$. A convolutional neural network (CNN) (Goodfellow et al., 2016) with parameters $\boldsymbol{\phi}$ can extract cost features from the environment map: $c_t(\boldsymbol{x},\boldsymbol{u}) = \mathbf{CNN}(\boldsymbol{h}_t,\boldsymbol{x},\boldsymbol{u};\boldsymbol{\phi})$. We implement an encoder-decoder neural network architecture (Badrinarayanan et al., 2017) to parameterize the cost function from semantic class probabilities. The idea is to perform downsamples and upsamples at multiple scales to provide both local and global context between semantic probability and cost.

4. Cost Learning via Differentiable Planning

We follow the planning algorithm in Wang et al. (2020) that enables efficient cost optimization and briefly review the steps below. The parameters $\boldsymbol{\theta}$ of the cost representation $c_t(\boldsymbol{x},\boldsymbol{u};\boldsymbol{P}_{1:t},\boldsymbol{\theta})$ developed in Sec. 3 are optimized by differentiating $\mathcal{L}(\boldsymbol{\theta})$ in (2) through the DSP problem in (1). Motion planning algorithms, such as A* (Likhachev et al., 2004), solve problem (1) efficiently and determine the optimal cost-to-go $Q_t^*(\boldsymbol{x},\boldsymbol{u})$ only over a subset of promising states. This is sufficient to obtain the subgradient of $Q_t^*(\boldsymbol{x}_t,\boldsymbol{u}_t)$ with respect to c_t along the optimal path by applying the subgradient method (Shor, 2012; Ratliff et al., 2006).

A backwards A* search applied to problem (1) with start state $\boldsymbol{x}_g$, goal state $\boldsymbol{x} \in \mathcal{X}$, and predecessors expansions according to transition f provides an upper bound to the optimal cost-to-go: $Q_t^*(\boldsymbol{x},\boldsymbol{u}) \leq c_t(\boldsymbol{x},\boldsymbol{u}) + g(f(\boldsymbol{x},\boldsymbol{u}))$, where g are the values computed by A* for expanded nodes in the CLOSED list and visited nodes in the OPEN list. Strict equality is obtained only if $f(\boldsymbol{x},\boldsymbol{u})$ belongs to the CLOSED list. A Boltzmann policy $\pi_t(\boldsymbol{u} \mid \boldsymbol{x})$ may be defined using the g-values for all $\boldsymbol{x} \in \text{CLOSED} \cup \text{OPEN} \subseteq \mathcal{X}$ and a uniform distribution over $\mathcal{U}$ for all other states.

Algorithm 1: Train cost parameters $\boldsymbol{\theta}$

input : $\mathcal{D} = \{(\boldsymbol{x}_{t,n}, \boldsymbol{u}^*_{t,n}, \boldsymbol{P}_{t,n}, \boldsymbol{x}_{g,n})\}_{t=1,n=1}^{T_n,N}$
while $\boldsymbol{\theta}$ *not converged* **do**
  $\mathcal{L}(\boldsymbol{\theta}) \leftarrow 0$;
  for $n = 1, \ldots, N$ ***and*** $t = 1, \ldots, T_n$ **do**
    Update $c_{t,n}$ based on $\boldsymbol{x}_{t,n}$ and $\boldsymbol{P}_{t,n}$ as in Sec. 3;
    Obtain $Q^*_{t,n}(\boldsymbol{x}, \boldsymbol{u})$ from (1) with stage cost $c_{t,n}$;
    Obtain $\pi_{t,n}(\boldsymbol{u}|\boldsymbol{x}_{t,n})$ from $Q^*_{t,n}(\boldsymbol{x}_{t,n}, \boldsymbol{u})$;
    $\mathcal{L}(\boldsymbol{\theta}) \leftarrow \mathcal{L}(\boldsymbol{\theta}) - \log \pi_{t,n}(\boldsymbol{u}^*_{t,n}|\boldsymbol{x}_{t,n})$;
  end
  Update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \nabla \mathcal{L}(\boldsymbol{\theta})$ via Prop. 4;
end

Algorithm 2: Test control policy π_t

input : Start state $\boldsymbol{x}_s$, goal state $\boldsymbol{x}_g$, cost parameters $\boldsymbol{\theta}$

Current state $\boldsymbol{x}_t \leftarrow \boldsymbol{x}_s$;
while $\boldsymbol{x}_t \neq \boldsymbol{x}_g$ **do**
  Make an observation $\boldsymbol{P}_t$;
  Update c_t based on $\boldsymbol{x}_t$ and $\boldsymbol{P}_t$ as in Sec. 3;
  Obtain $\pi_t(\boldsymbol{u}|\boldsymbol{x}_t)$ from $Q^*_t(\boldsymbol{x}_t, \boldsymbol{u})$;
  Update $\boldsymbol{x}_t \leftarrow f(\boldsymbol{x}_t, \boldsymbol{u}_t)$ via $\boldsymbol{u}_t := \arg\max_{\boldsymbol{u}} \pi_t(\boldsymbol{u}|\boldsymbol{x}_t)$;
end

We rewrite $Q^*_t(\boldsymbol{x}_t, \boldsymbol{u}_t)$ in a form that makes its subgradient with respect to $c_t(\boldsymbol{x}, \boldsymbol{u})$ obvious. Let $\mathcal{T}(\boldsymbol{x}_t, \boldsymbol{u}_t)$ be the set of feasible trajectories $\boldsymbol{\tau}$ of horizon T that start at $\boldsymbol{x}_t$, $\boldsymbol{u}_t$, satisfy transition f and terminate at $\boldsymbol{x}_g$. Let $\boldsymbol{\tau}^* \in \mathcal{T}(\boldsymbol{x}_t, \boldsymbol{u}_t)$ be an optimal trajectory corresponding to the optimal cost-to-go function $Q^*_t(\boldsymbol{x}_t, \boldsymbol{u}_t)$. Define $\mu_{\boldsymbol{\tau}}(\boldsymbol{x}, \boldsymbol{u}) := \sum_{k=t}^{T-1} \mathbb{1}_{(\boldsymbol{x}_k, \boldsymbol{u}_k) = (\boldsymbol{x}, \boldsymbol{u})}$ as a state-control visitation function indicating if $(\boldsymbol{x}, \boldsymbol{u})$ is visited by $\boldsymbol{\tau}$. The optimal cost-to-go function $Q^*_t(\boldsymbol{x}_t, \boldsymbol{u}_t)$ can be viewed as a minimum over $\mathcal{T}(\boldsymbol{x}_t, \boldsymbol{u}_t)$ of the inner product of the cost function c_t and the visitation function $\mu_{\boldsymbol{\tau}}$:

$$Q^*_t(\boldsymbol{x}_t, \boldsymbol{u}_t) = \min_{\boldsymbol{\tau} \in \mathcal{T}(\boldsymbol{x}_t, \boldsymbol{u}_t)} \sum_{\boldsymbol{x} \in \mathcal{X}, \boldsymbol{u} \in \mathcal{U}} c_t(\boldsymbol{x}, \boldsymbol{u}) \mu_{\boldsymbol{\tau}}(\boldsymbol{x}, \boldsymbol{u}) \tag{8}$$

where $\mathcal{X}$ can be assumed finite because both T and $\mathcal{U}$ are finite. Applying the subgradient method (Shor, 2012; Ratliff et al., 2006) to (8) shows that $\frac{\partial Q^*_t(\boldsymbol{x}_t, \boldsymbol{u}_t)}{\partial c_t(\boldsymbol{x}, \boldsymbol{u})} = \mu_{\boldsymbol{\tau}^*}(\boldsymbol{x}, \boldsymbol{u})$ is a subgradient of the optimal cost-to-go. This result and the chain rule allow us to obtain a subgradient of $\mathcal{L}(\boldsymbol{\theta})$.

Proposition 4 *A subgradient of the loss function $\mathcal{L}(\boldsymbol{\theta})$ in (2) with respect to $\boldsymbol{\theta}$ can be obtained as:*

$$\frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = -\sum_{n=1}^{N} \sum_{t=1}^{T_n} \frac{\partial \log \pi_{t,n}(\boldsymbol{u}^*_{t,n} \mid \boldsymbol{x}_{t,n})}{\partial \boldsymbol{\theta}} = -\sum_{n=1}^{N} \sum_{t=1}^{T_n} \sum_{\boldsymbol{u}_{t,n} \in \mathcal{U}} \frac{\partial \log \pi_{t,n}(\boldsymbol{u}^*_{t,n} \mid \boldsymbol{x}_{t,n})}{\partial Q^*_{t,n}(\boldsymbol{x}_{t,n}, \boldsymbol{u}_{t,n})} \frac{\partial Q^*_{t,n}(\boldsymbol{x}_{t,n}, \boldsymbol{u}_{t,n})}{\partial \boldsymbol{\theta}}$$

$$= -\sum_{n=1}^{N} \sum_{t=1}^{T_n} \sum_{\boldsymbol{u}_{t,n} \in \mathcal{U}} \left(\mathbb{1}_{\{\boldsymbol{u}_{t,n} = \boldsymbol{u}^*_{t,n}\}} - \pi_{t,n}(\boldsymbol{u}_{t,n}|\boldsymbol{x}_{t,n}) \right) \sum_{(\boldsymbol{x}, \boldsymbol{u}) \in \boldsymbol{\tau}^*} \frac{\partial Q^*_{t,n}(\boldsymbol{x}_{t,n}, \boldsymbol{u}_{t,n})}{\partial c_t(\boldsymbol{x}, \boldsymbol{u})} \frac{\partial c_t(\boldsymbol{x}, \boldsymbol{u})}{\partial \boldsymbol{\theta}}$$

The computation graph implied by Prop. 4 is illustrated in Fig. 1. The graph consists of a cost representation layer and a differentiable planning layer, allowing end-to-end minimization of $\mathcal{L}(\boldsymbol{\theta})$ via stochastic subgradient descent. Training and testing algorithms are shown in Alg. 1 and Alg. 2.

5. Experiments

5.1. Experiment Setup

We evaluate our approach using the CARLA simulator (0.9.6) (Dosovitskiy et al., 2017), which provides high-fidelity autonomous vehicle simulation in urban environments. Demonstration data for training the cost function representation is collected from maps $\{Town01, Town02, Town03, Town04\}$, while map $Town05$ is used for testing. $Town05$ is the largest map and includes different

street layouts, junctions, and a freeway. In each map, we collect 100 expert trajectories by running the autonomous navigation agent provided by the CARLA Python API. The expert finds the shortest path between two query points, while respecting traffic rules, such as staying on the road, and keeping in the current lane. Features not related to the experiment are disabled, including spawning other vehicles and pedestrians, and changing traffic signal. Each vehicle trajectory is discretized into a 128×128 grid of 1 meter resolution. The robot state $\boldsymbol{x}$ is the grid cell location while the control $\boldsymbol{u}$ takes the robot to one of its 8 neighbor grid cells. Trajectories that do not fit in the grid are discarded.

The ego vehicle is equipped with a lidar sensor that has 20 meters maximum range and 360° horizontal field of view. The vertical field of view ranges from 0° (facing forward) to -30° (facing down) with 5° resolution. A total of 56000 lidar rays is generated per scan $\boldsymbol{P}_t$ while each point measurement is returned only if it hits an obstacle. The ego vehicle is also equipped with 4 semantic segmentation cameras that display objects of 13 different classes in RGB images, including road, road line, sidewalk, vegetation, car, building, etc. The 4 cameras face front, left, right and rear, each capturing a 90° horizontal field of view. The semantic label of each lidar point is retrieved from the semantic segmentation image by projecting the lidar point in the camera's frame.

5.2. Models and Metrics

We compare our model with two baseline algorithms: Wulfmeier et al. (2016) and Wulfmeier et al. (2016) + semantics. Wulfmeier et al. (2016) use a neural network to learn a cost from lidar point clouds without semantics. The input to the neural network is a grid that stores the mean and variance of points in each cell, as well as a binary indicator of cell visibility. We augment the grid features with the mode of semantic labels in each cell to get the model Wulfmeier et al. (2016) + semantics as a fair comparison with ours. Neural networks are implemented in the PyTorch library (Paszke et al., 2019) and trained with the Adam optimizer (Kingma and Ba, 2014) until convergence.

The evaluation metrics include: negative log-likelihood (*NLL*), control accuracy (*Acc*), trajectory success rate (*Traj. Succ. Rate*) and Modified Hausdorff Distance (*MHD*). More precisely, given a test set $\mathcal{D}_{test} = \left\{(\boldsymbol{x}_{t,n}, \boldsymbol{u}^*_{t,n}, \boldsymbol{P}_{t,n}, \boldsymbol{x}_{g,n})\right\}_{t=1,n=1}^{T_n,N}$ and a learned policy π with paramters $\boldsymbol{\theta}^*$, we define $NLL(\mathcal{D}_{test}, \pi) = -\frac{1}{\sum_{n=1}^{N} T_n} \sum_{n=1,t=1}^{N,T_n} \log \pi_{t,n}(\boldsymbol{u}^*_{t,n} | \boldsymbol{x}_{t,n}; \boldsymbol{P}_{1:t,n}, \boldsymbol{\theta}^*)$ and $Acc(\mathcal{D}_{test}, \pi) = \frac{1}{\sum_{n=1}^{N} T_n} \sum_{n=1,t=1}^{N,T_n} \mathbb{1}_{\left\{\boldsymbol{u}^*_{t,n} = \arg\max \pi_{t,n}(\cdot | \boldsymbol{x}_{t,n}; \boldsymbol{P}_{1:t,n}, \boldsymbol{\theta}^*)\right\}}$. *Traj. Succ. Rate* records the success rate of the learned policy by iteratively rolling out its predicted controls. A trajectory is regarded as successful if it reaches the goal within twice the number of steps of the expert trajectory without hitting an obstacle. *MHD* compares the rolled out trajectory $\boldsymbol{\tau}_L$ by the learned policy and the expert trajectory $\boldsymbol{\tau}_E$ and is defined as: $MHD(\boldsymbol{\tau}_L, \boldsymbol{\tau}_E) = \max\left\{\frac{1}{T_L}\sum_{t=1}^{T_L} d(\boldsymbol{\tau}^t_L, \boldsymbol{\tau}_E), \frac{1}{T_E}\sum_{t=1}^{T_E} d(\boldsymbol{\tau}^t_E, \boldsymbol{\tau}_L)\right\}$ where $d(\boldsymbol{\tau}^t_A, \boldsymbol{\tau}_B)$ measures the minimum Euclidean distance from state $\boldsymbol{\tau}^t_A$ to any state in $\boldsymbol{\tau}_B$.

5.3. Results and Discussion

Fig. 2 shows the performance of our model versus Wulfmeier et al. (2016) and Wulfmeier et al. (2016) + semantics using the metrics described above. Ours learns to generate policies closest to the expert in new environments by scoring best in *NLL* and *Acc*. The predicted trajectory is also closest to the expert by achieving the minimum *MHD*. The results demonstrate that the semantic map encoder captures more geometric as well as semantic information so that the cost function can

Model	*NLL*	*Acc* (%)	*Traj. Succ. Rate* (%)	*MHD*
Wulfmeier et al. (2016)	0.595	86.1	**92**	4.521
Wulfmeier et al. (2016) + semantics	0.613	82.7	88	4.479
Ours	**0.446**	**90.5**	91	**3.036**

Figure 2: Test result from CARLA *Town05* map. Best model for each evaluation metric is in bold.

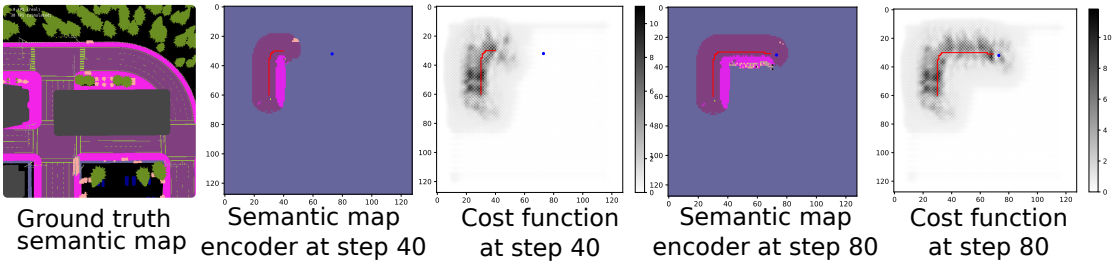

Figure 3: Example of a predicted trajectory in red at an intersection and the goal in blue. The groud truth semantic map, predicted semantic map and cost map at two time steps are shown. Our model learns the sidewalk is costly to traverse.

be optimized and generate trajectories which match the expert behaviors. We notice that simply taking the mode of the semantic labels in each grid cell degrades the performance of Wulfmeier et al. (2016). We conjecture that taking the mode is a deterministic assignment, which could provide conflicting semantic information, while our model endorses a probabilistic semantic map encoder with Bayesian updates to avoid information loss. Fig. 3 shows an example of the predicted trajectory at an intersection. The semantic map visualizes the class of highest probability, which mostly reflects the ground truth. Sub-cell objects like roadlines are captured in the semantic map distribution but not visualized in the most probable class. It is interesting to find that our model assigns low cost to road in front of the robot, medium cost for sidewalks, and high cost to road behind itself. This cost assignment is actually effective for the robot to navigate to the goal.

6. Conclusion

We propose an inverse reinforcement learning approach for infering navigation costs from demonstrations with semantic observations. Our model introduces a new cost representation composed of a probabilistic semantic occupancy encoder and a cost encoder defined over the semantic features. The cost function can be optimized via backpropagation with closed-form (sub)gradient. Experiments in the CARLA simulator show that our model outperforms methods that do not encode semantic information probabilistically over time. Our work offers a promising solution for learning semantic features in navigation and may enable efficient online learning in challenging conditions.

Acknowledgments

We gratefully acknowledge support from NSF CRII IIS-1755568, ARL DCIST CRA W911NF-17-2-0181, and ONR SAI N00014-18-1-2828.

References

Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39 (12):2481–2495, 2017.

Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.

David Dohan, Brian Matejek, and Thomas Funkhouser. Learning hierarchical semantic segmentations of lidar data. In *2015 International Conference on 3D Vision*, pages 273–281. IEEE, 2015.

Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International conference on machine learning*, pages 49–58, 2016.

Lu Gan, Ray Zhang, Jessy W Grizzle, Ryan M Eustice, and Maani Ghaffari. Bayesian spatial kernel smoothing for scalable dense semantic mapping. *arXiv preprint arXiv:1909.04631*, 2019.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik. Cognitive Mapping and Planning for Visual Navigation. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

Armin Hornung, Kai M Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 34(3):189–206, 2013.

A. Khan, C. Zhang, N. Atanasov, K. Karydis, V. Kumar, and D. D. Lee. Memory augmented control networks. In *International Conference on Learning Representations (ICLR)*, 2018.

Diederik P Kingma and Jimmy Ba. ADAM: A method for stochastic optimization. In *ICLR*, 2014.

Sergey Levine, Zoran Popovic, and Vladlen Koltun. Nonlinear inverse reinforcement learning with gaussian processes. In *Advances in Neural Information Processing Systems*, pages 19–27, 2011.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *J. Mach. Learn. Res.*, 17(1):1334–1373, 2016.

M. Likhachev, G. Gordon, and S. Thrun. ARA*: Anytime A* with Provable Bounds on Sub-Optimality. In *in Advances in Neural Information Processing Systems*, page 767774, 2004.

C. Lu, M. J. G. van de Molengraft, and G. Dubbelman. Monocular semantic occupancy grid mapping with convolutional variational encoderdecoder networks. *IEEE Robotics and Automation Letters*, 4(2):445–452, April 2019. ISSN 2377-3774. doi: 10.1109/LRA.2019.2891028.

Gergely Neu and Csaba Szepesvári. Apprenticeship learning using inverse reinforcement learning and gradient methods. In *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence*, pages 295–302. AUAI Press, 2007.

H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto. Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1366–1373, 2017. doi: 10.1109/IROS.2017.8202315.

George Papandreou, Liang-Chieh Chen, Kevin P Murphy, and Alan L Yuille. Weakly-and semi-supervised learning of a deep convolutional network for semantic image segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1742–1750, 2015.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. In *IJCAI*, volume 7, pages 2586–2591, 2007.

Nathan D Ratliff, J Andrew Bagnell, and Martin A Zinkevich. Maximum margin planning. In *Proceedings of the 23rd international conference on Machine learning*, pages 729–736. ACM, 2006.

Sunando Sengupta, Paul Sturgess, L'ubor Ladickỳ, and Philip HS Torr. Automatic dense visual semantic mapping from street-level imagery. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 857–862. IEEE, 2012.

Naum Zuselevich Shor. *Minimization methods for non-differentiable functions*, volume 3. Springer Science & Business Media, 2012.

Yeeho Song. Inverse reinforcement learning for autonomous ground navigation using aerial and satellite observation data. Master's thesis, Pittsburgh, PA, August 2019.

L. Sun, Z. Yan, A. Zaganidis, C. Zhao, and T. Duckett. Recurrent-octomap: Learning state-based map refinement for long-term semantic mapping with 3-d-lidar data. *IEEE Robotics and Automation Letters*, 3 (4):3749–3756, Oct 2018. ISSN 2377-3774.

Aviv Tamar, Yi Wu, Garrett Thomas, Sergey Levine, and Pieter Abbeel. Value iteration networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2154–2162. Curran Associates, Inc., 2016. URL http://papers.nips.cc/paper/6046-value-iteration-networks.pdf.

Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. The MIT Press, 2005. ISBN 0262201623.

Tianyu Wang, Vikas Dhiman, and Nikolay Atanasov. Learning navigation costs from demonstration in partially observable environments. In *2020 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2020.

Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1887–1893. IEEE, 2018.

Markus Wulfmeier, Dominic Zeng Wang, and Ingmar Posner. Watch this: Scalable cost-function learning for path planning in urban environments. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2089–2095. IEEE, 2016.

Brian D. Ziebart, Andrew Maas, J.Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, pages 1433–1438, 2008.