Check for updates

Article

The International Journal of Robotics Research
1073–1087
© The Author(s) 2017
Reprints and permissions:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/0278364917722396
journals.sagepub.com/home/ijr

Large-scale cost function learning for path planning using deep inverse reinforcement learning

Markus Wulfmeier, Dushyant Rao, Dominic Zeng Wang, Peter Ondruska and Ingmar Posner

Abstract

We present an approach for learning spatial traversability maps for driving in complex, urban environments based on an extensive dataset demonstrating the driving behaviour of human experts. The direct end-to-end mapping from raw input data to cost bypasses the effort of manually designing parts of the pipeline, exploits a large number of data samples, and can be framed additionally to refine handcrafted cost maps produced based on manual hand-engineered features. To achieve this, we introduce a maximum-entropy-based, non-linear inverse reinforcement learning (IRL) framework which exploits the capacity of fully convolutional neural networks (FCNs) to represent the cost model underlying driving behaviours. The application of a high-capacity, deep, parametric approach successfully scales to more complex environments and driving behaviours, while at deployment being run-time independent of training dataset size. After benchmarking against state-of-the-art IRL approaches, we focus on demonstrating scalability and performance on an ambitious dataset collected over the course of 1 year including more than 25,000 demonstration trajectories extracted from over 120 km of urban driving. We evaluate the resulting cost representations by showing the advantages over a carefully, manually designed cost map and furthermore demonstrate its robustness towards systematic errors by learning accurate representations even in the presence of calibration perturbations. Importantly, we demonstrate that a manually designed cost map can be refined to more accurately handle corner cases that are scarcely seen in the environment, such as stairs, slopes and underpasses, by further incorporating human priors into the training framework.

Keywords

Learning from demonstration, inverse reinforcement learning, neural networks, autonomous driving, cost functions

1. Introduction

A mission-critical capability for an autonomous vehicle is determining the traversability of its surroundings and computing a safe trajectory to its goal. Typically, this is performed by exploiting manually designed cost functions, which specify the cost of visiting each location or state as a function of sensory input data. However, the mapping between sensor input and cost is likely to be very complex. In particular, in environments containing a wide variety of obstacles, such as stairs, bollards, underpasses, or slopes, the cost of traversing a particular location is a complex function of the sensor data, and is additionally dependent on features in the spatial neighbourhood, rather than just the location itself. Consequently, the manual handcrafting of cost functions for motion planning is a difficult and laborious task, requiring expertise in robotics, sensing and motion planning.

Learning from demonstration (LfD), also known as programming by demonstration, is intended to render the task of specifying robot behaviour independent of knowledge in robotics and applied algorithms. The approach is often employed to enable untrained personnel to train and adapt machines to solve tasks via imitation.

The inverse reinforcement learning (IRL) framework enables this paradigm and centers on learning the underlying reward or cost structure from demonstrations of human behaviour (Ziebart et al., 2008). It has been applied to a wide range of domains, including autonomous driving (Wulfmeier et al., 2016a), robotic manipulation (Finn et al., 2016) and grid-world planning (Nguyen et al., 2015). In principle, IRL eliminates the expertise and effort required to manually design cost functions, and can produce cost maps that are more consistent with human behaviour. However, the original IRL framework utilises a linear mapping

Oxford Robotics Institute, University of Oxford, UK

Corresponding author:
Markus Wulfmeier, Oxford Robotics Institute, University of Oxford, Parks Road Oxford OX1 3PJ, UK.
Email: markus@robots.ox.ac.uk

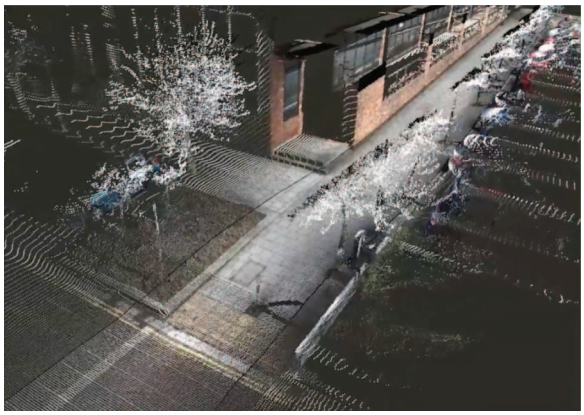


Fig. 1. Coloured 3D LIDAR environment scan.

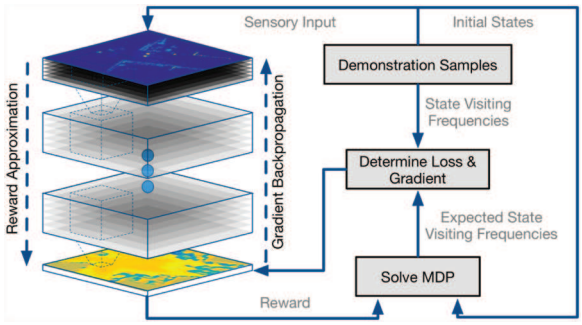


Fig. 2. Schema for training neural networks in the maximum entropy paradigm for IRL.

between input features and output reward, which severely restricts the complexity of cost structures that can be modelled accurately. Other state-of-the-art methods that have the required capacity are built upon non-parametric models such as Gaussian processes (GPs), which have a runtime dependent on the size of the dataset and are quickly rendered intractable for large amounts of demonstration data (Levine et al., 2011; Ramachandran and Amir, 2007).

In recent years, the enormous expressive capacity of deep neural networks has led to state-of-the-art performance in numerous applications and tasks (Liu et al., 2015; Long et al., 2015; Sermanet et al., 2010). We posit that as robust, flexible function approximators they are well-suited to an IRL framework. In particular, they have the capacity to model the complex relationship between sensory input and reward structure, are able to capture spatial correlations in the data using convolutional layers and exhibit constant time operation, allowing them to scale to very large datasets.

Our recent work explores the use of fully convolutional neural networks (FCNs) to model the reward structure for a synthetic dataset (Wulfmeier et al., 2015), and for a real-world application to LIDAR data from a mobile vehicle (Wulfmeier et al., 2016a). The work demonstrated the ability of CNNs to accurately infer the underlying reward structure from human driving demonstrations, leading to control policies that more closely mimicked the behaviour of a human driver. This was further extended in Wulfmeier et al. (2016b) with an additional pretraining stage to further improve the overall accuracy of the model and its performance on corner cases, such as planning near slopes, bollards and stairs. This journal article presents a coherent framework for large-scale, non-linear IRL which unifies and extends our prior work by including more detailed evaluation and discussion.

The principal contributions of our work are as follows.

- Development of Maximum Entropy Deep Inverse Reinforcement Learning (MEDIRL), a framework for applying high-capacity neural network architectures to solve the IRL problem. This extends scalability with respect to complexity of environment, behaviour and size of training datasets.
- Qualitative and quantitative evaluation of the approach on small-scale scenarios against state-of-the-art approaches in IRL. We experimentally validate the convergence of task-optimal spatial features through IRL-based training given enough demonstration samples on common benchmark scenarios.
- Evaluation of scalability and efficacy in real-world tasks on an extensive dataset of over 120 km capturing the common driving behaviour of 13 different human drivers, resulting in over 25,000 training samples.
- Demonstration of robustness towards systematic noise in form of miscalibration of the sensor setup and demonstrating significantly stronger performance for prediction of human trajectories and classification of traversable terrain.
- Refining human prior cost maps to address additional complexities of real-world sensor data and sparse feedback from IRL to combine benefits of human intuition with large-scale demonstration datasets.

2. Related work

Manual cost function design builds a foundation for the majority of state-of-the-art motion planning systems for autonomous driving (e.g. Choset, 2005), with recent successful examples given by the competing teams in the DARPA Grand (Thrun et al., 2006) and Urban Challenges (Montemerlo et al., 2008; Urmson et al., 2009). Obstacles have to be explicitly identified and are typically inflated as a function of the vehicle size. The necessary weighting of costs from different sensing modalities and the adaptation for different driving behaviours relies on detailed domain knowledge. In addition, the task of extracting good features from raw input data for computing the cost maps is often non-trivial and relies heavily on a well-calibrated hardware setup.

Manually designing these cost functions can prove challenging and time demanding, and limits the task to highly specialised personnel. In order to deploy autonomous mobile vehicles in new locations and open the task to untrained personnel, LfD approaches represent a promising

alternative and have led to recent advances in principally two different areas, direct policy imitation and IRL. Policy imitation, also commonly known as behavioural cloning, targets directly learning the policy mapping from perceived environment or preprocessed features to the agent's actions, while IRL in contrast focuses on inferring the agent's underlying reward structure: the preferences that cause specific behaviours. While we mention a select number of relevant works here, the interested reader is referred to Argall et al. (2009) for a detailed summary.

Early work to directly model driving behaviour with neural networks (Pomerleau, 1990) focused on learning a discretised steering output based on downsampled front-facing image streams and fully connected layers in the neural network. This approach was extended in recent work via the application of convolutional neural networks (CNNs) (Chen et al., 2015; Sermanet et al., 2010) and scaled significantly with respect to dataset size, task complexity and performance.

While direct policy imitation is well suited for the design of reactive controllers, long-term decision-making systems need to plan further ahead when aiming to find a safe, collision-free path in possibly cluttered terrains. The described principal challenge for policy LfD lies in generalisation as policies are learned along trajectories and deviations from those introduce large errors. Ross et al. (2010) address this problem by querying an expert to improve the policy for states not encountered in the original trajectories but this requires continuous human supervision and monitoring.

Furthermore, especially in safety critical situations, where future behaviour has to be tested against specific constraints, the capacity to generate long-term plans rather than instantaneous reactions is necessary to ensure robust and safe interaction with the environment. A reward model is generally seen to be more succinct than the policy and conceived to be preferable as generalisation becomes important (Abbeel and Ng, 2004; Abbeel et al., 2008). While a policy is directed towards solving a specific task, a cost-function-based approach can more easily be extended by switching the goal and deriving a new policy for the setting.

Owing to its strengths in planning tasks and the capability for integration into existing systems, recent work successfully applies IRL to planning (Ratliff et al., 2009). Further applications include improving driving and robotic navigation with focus on interaction of mobile autonomous platforms and humans (Kuderer et al. 2015; Kretzschmar et al. 2016; Pfeiffer et al. 2016). These works additionally target the modelling of the behaviour of other participants, which represents another principal application for IRL approaches.

Early work in IRL started off with linear parametrisation of the reward function based on manually pre-determined features (Abbeel and Ng, 2004; Lopes et al., 2009; Ratliff et al., 2006; Ziebart et al., 2008). In order to overcome the inherent limitations of linear models, following approaches extended this approach to a limited set of non-linear rewards and learning to build composites of logical conjunctions for atomic features (Choi and Kim, 2013; Levine et al., 2010).

More flexible non-linear function approximators such as GPs further extend the modelling capacity of IRL models (Levine et al., 2011). However, the application of a non-parametric approach causes the approach to scale poorly with the numbers of demonstration samples, quickly rendering the GP framework impractical due to the unfavourable computational complexity. Even sparse GP approximations lead to training computation complexity that is dependent on the size of the active set and the number of encountered samples ($\mathcal{O}(n \times m^2)$ where n denotes the number of experienced state–reward pairs and m the number of inducing points). Therefore, situations with increasingly complex reward functions, behaviours or environments that lead to higher requirements regarding the number of inducing points will render a non-parametric approach impracticable.

To address these challenges while maintaining the accuracy of high capacity, non-linear function approximation, we introduce deep parametric models, in particular FCNs, and develop a framework based on maximum entropy (MaxEnt) IRL (Ziebart et al., 2008), an approach that constrains the distribution of demonstration trajectories to the one of highest possible entropy. The resulting objective function is differentiable with respect to the network weights based on feature matching (Abbeel and Ng, 2004) and backpropagation as illustrated in detail in Section 3.3. The approach enables the application of arbitrary neural network architectures in the context of IRL. Recent successful examples for the application of FCNs can be found in pixel-wise semantic segmentation (Long et al., 2015), sliding window detection and prediction of object boundaries (Sermanet et al., 2013) and depth estimation with single monocular images (Liu et al., 2015) .

In parallel, Finn et al. (2016) developed a sampling-based approach to IRL for training neural networks with only fully connected layers in a MaxEnt-IRL-based framework for robotic manipulation and navigation tasks. The cost mapping is determined from crafted feature representations that are parametrised based on the trajectory representation. The approach requires sampling of real-world training data during the training process to approximate the complex dynamic model. In contrast, our work focuses on employing *fully convolutional networks* in large-scale motion planning problems such as autonomous driving, and does not require run-time sampling. Being model-based, our proposed method removes the requirement of sampling on the actual platform resulting in increased safety and streamlining of the training procedure.

3. Problem formulation

When employing an autonomous vehicle in a new environment, it is pivotal to incorporate knowledge about terrain traversability into the system. While it is possible to

describe generally reasonable cost functions to capture the terrain constraints, it is very difficult and time consuming to consider all possible corner cases. In the following section, we present an approach to automatically learn these cost functions for large datasets of demonstration trajectories in the context of autonomous driving.

3.1. IRL

To overcome the limitations of manual handcrafting of cost functions, we frame the aforementioned task in the context of IRL, which has the principal goal of inferring the preferences or the reward structure that underlies specific behaviours. 'Reward' and 'cost' will be used interchangeably in the following sections as one is the straightforward negation of the other.

The approach is commonly formulated using the Markov decision process framework $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, T, \gamma, r\}$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ denotes the set of possible actions, T is the state transition model, γ is a discount factor that modulates the influence of future rewards and $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is a function specifying the reward structure. As r is not provided, it must be inferred from a set of demonstrations $\mathcal{D} = \{\varsigma_1, \varsigma_2, \ldots, \varsigma_N\}$, each of which is a sequence of state–action pairs $\varsigma_i = \{(s_1, a_1), (s_2, a_2), \ldots, (s_K, a_K)\}$ representing a sample trajectory.

3.2. MaxEnt IRL

When solving the IRL problem, one inherently needs to address two main challenges: suboptimality of demonstration trajectories given the underlying reward, as no human expert will ever act completely optimally; and reward ambiguity, as multiple rewards can explain the same behaviour.

The MaxEnt approach addresses these challenges by modelling expert behaviour as a distribution over trajectories and constraining this distribution to the one of highest entropy (Ziebart et al., 2008). It defines the demonstrator's underlying policy $\pi_D(a|s)$ such that the probability for user preference of any given trajectory between specified start and goal states is proportional to the exponential of the reward along the path:

$$P(\varsigma \mid r) = \prod_{i=1}^{K} \pi_D(a_i \mid s_i) \propto \exp\left\{\sum_{i=1}^{K} r_{s_i,a_i}\right\} \tag{1}$$

with trajectory length K. This approach results in the least biased estimate given the demonstration trajectories (Zhifei and Joo, 2012).

3.3. MEDIRL

In the original MaxEnt IRL formulation (Ziebart et al., 2008) the reward r was defined as a linear function of input features $\boldsymbol{f}$, i.e. $r = \boldsymbol{\theta}^T \boldsymbol{f}$. The features in this context are the observations at a given state $f(s)$ such that the reward is parametrised on sensor observations rather than absolute state, which supports generalisation and enables application in large state spaces.

The application of a linear model relies on the ability to extract meaningful features in the data, and restricts the complexity of the mapping from input to reward that can be learned. Instead this approach moves all complexity into the manual design of optimal features.

In this work, we instead propose MEDIRL which employs FCNs to model the reward structure directly as a non-linear function of the inputs. Neural networks are flexible, powerful, function approximators, which makes them ideal for this task of predicting the reward structure for a given sensor input. By utilising convolutional layers for this task, we eliminate the need to handcraft spatial feature mappings from raw data input, which instead can be learned to be optimal for the respective task.

The network is trained to maximise the joint probability of the demonstration data and model parameters under the predicted reward:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}) &= \log P(\mathcal{D}, \boldsymbol{\theta} \mid r(\boldsymbol{\theta})) \\ &= \underbrace{\log P(\mathcal{D} \mid r(\boldsymbol{\theta}))}_{\mathcal{L}_\mathcal{D}} + \underbrace{\log P(\boldsymbol{\theta})}_{\mathcal{L}_\theta} \end{aligned} \tag{2}$$

where the loss function can be split into the log-likelihood of the demonstration data $\mathcal{L}_\mathcal{D}$, and a regularisation term denoting the log probability of the network parameters $\mathcal{L}_\theta$. Here, we will focus on the data likelihood since parameter regularisation terms can build on various, common approaches such as L1-norm and L2-norm or dropout (Hinton et al., 2012).

The data likelihood term relates to differences in feature counts $\left(\boldsymbol{f}_\mathcal{D} - \mathbb{E}[\boldsymbol{f}]\right)$ (Abbeel and Ng, 2004). In the original MaxEnt formulation (Ziebart et al., 2008), this is specified as a linear model, with the differences in state visitation frequencies $(\boldsymbol{\mu}_\mathcal{D} - \mathbb{E}[\boldsymbol{\mu}])$ multiplied by a matrix $\boldsymbol{F}_s$, as in Equation (5). Here, $\boldsymbol{\mu}_\mathcal{D}$ refers to the mean state visitation frequencies according to the demonstration trajectories, and $\mathbb{E}[\boldsymbol{\mu}]$ is the expected state visitation frequencies under the policy. Thus, both feature and state visitation difference terms in this context express how much the behaviour of the learned model differs from the behaviour underlying the demonstration trajectories:

$$\frac{\partial \mathcal{L}_\mathcal{D}}{\partial \boldsymbol{\theta}} = \frac{\partial \mathcal{L}_\mathcal{D}}{\partial r} \frac{\partial r}{\partial \boldsymbol{\theta}} \tag{3}$$

$$= \boldsymbol{f}_\mathcal{D} - \mathbb{E}[\boldsymbol{f}] \tag{4}$$

Linear MaxEnt IRL

$$= (\boldsymbol{\mu}_\mathcal{D} - \mathbb{E}[\boldsymbol{\mu}])\, \boldsymbol{F}_s \tag{5}$$

MEDIRL

$$= \underbrace{(\boldsymbol{\mu}_\mathcal{D} - \mathbb{E}[\boldsymbol{\mu}])}_{\text{State Visitation Matching}} \underbrace{\partial r(\boldsymbol{\theta}) / \partial \boldsymbol{\theta}}_{\text{Backpropagation}} . \tag{6}$$

By extending the original linear formulation to neural networks, we benefit from the efficiency of gradient backpropagation, expressed in the second term in Equation (6), to

Algorithm 1 MEDIRL

Input: $\boldsymbol{\mu}_D^a, \boldsymbol{f}, \mathcal{S}, \mathcal{A}, T, \gamma, \alpha$
Output: network parameters $\boldsymbol{\theta}^*$

1: **for** n = 1 : N **do**
2: $r^n = r(\boldsymbol{f}, \boldsymbol{\theta}^n)$

planning step
3: $\boldsymbol{\pi}^n = \text{approx_value_iteration}(r^n, \mathcal{S}, \mathcal{A}, T, \gamma)$
4: $\mathbb{E}[\boldsymbol{\mu}^n] = \text{propagate_policy}(\boldsymbol{\pi}^n, \mathcal{S}, \mathcal{A}, T)$

determine objective
5: $\mathcal{L}_D^n = \log(\boldsymbol{\pi}^n) \times \boldsymbol{\mu}_D^a$
6: $\frac{\partial \mathcal{L}_D^n}{\partial r^n} = \boldsymbol{\mu}_D - \mathbb{E}[\boldsymbol{\mu}^n]$

parameter update
7: $\frac{\partial \mathcal{L}_D^n}{\partial \boldsymbol{\theta}^n} = \frac{\partial \mathcal{L}_D^n}{\partial r^n} \frac{\partial r^n}{\partial \boldsymbol{\theta}^n}$
8: $\boldsymbol{\theta}^{n+1} = \boldsymbol{\theta}^n + \alpha \frac{\partial \mathcal{L}_D^n}{\partial \boldsymbol{\theta}^n}$
9: **end for**

Algorithm 2 Approximate Value Iteration

Input: $r^n, \mathcal{S}, \mathcal{A}, T, \gamma$
Output: π^n
1: $V(s) = -\infty$
2: **repeat**
3: $V_t(s) = V(s)\,; \;\; V(s_{goal}) = 0$
4: $Q(s,a) = r(s,a) + E_{T(s,a,s')}[V(s')]$
5: $V(s) = softmax_a \; Q_i(s,a)$
6: **until** $max_s(V(s) - V_t(s)) < \epsilon$
7: $\pi^n(a|s) = e^{Q(s,a)-V(s)}$

find the non-linear mapping from the difference in state visitation frequencies $\boldsymbol{\mu}$ to the necessary updates for all relevant parameters.

Algorithm 1 illustrates the steps for iteratively refining cost functions in the MEDIRL framework to maximise the probability of expert demonstration trajectories. In each step the current cost function is evaluated based on input observations $\boldsymbol{f}$ and the current parameters $\boldsymbol{\theta}^n$. Given the current cost function, lines 3 and 4 determine the state visiting frequencies based on Equation (1). Algorithm 2 describes a common method for computing approximate value iteration and determines the policy based on the current approximation of the reward and the transition model. Lines 5 and 6 subsequently compute the current objective and the gradient with respect to the reward, leading to the network parameter update in lines 7 and 8.

3.4. Architectures

A first comparison based on toy scenarios in Section 4.1 applies only simple, relatively shallow networks as they suffice for the task at hand. Furthermore, we evaluate three more advanced architectures with varying properties for the application of cost function learning for motion planning on a large-scale real-world dataset.

In addition to a standard, serial FCN, we introduce two variations to overcome the shortcomings of this basic implementation. We argue that the exact position of a feature is often less relevant than its general presence, as, for example, any obstacle will influence a larger surrounding group of states in the context of motion planning cost functions. In classification tasks this notion of limited translational invariance is commonly addressed via a max-pooling layer, leading to the implementation of the pooling FCN. This approach eliminates the need to learn specific filter kernels for all positions of relevant features and reduces the chance of overfitting as opposed to increasing the network capacity to learn all otherwise necessary filters. This property is particularly important for the following large-scale evaluation in Section 4.2 where, due to the sparsity of the LIDAR pointcloud depending on distance from the car, the feature representation for objects differs depending on where they are located. The approach reduces the need to learn all resulting feature patterns in this context.

While the pooling-based architecture leads to increased spatial invariance with the advantages mentioned earlier in this section, this also leads to the irreversible loss of location information for low-level features. In order to combine the benefits while maintaining the networks ability to utilise exact location information when needed, we introduce the multi-scale (MS) FCN architecture shown in Figure 3 that is related to the multi-scale deep jet architecture (Long et al., 2015) which integrates features of different scales for image segmentation. The proposed MS FCN architecture is able to treat the feature channels separate by concatenation instead of summation, similar to Szegedy et al. (2015). In contrast to the deep jet architecture, where feature channels from parallel branches share the same semantic meaning, we intend to learn independent filter kernels representing different factors of influence for our cost maps.

Furthermore, a benefit of keeping all architectures fully convolutional is that we can design the size of the receptive field for each location in the final cost map to ensure coverage for an area encapsulating the size of the vehicle and all factors that might influence driving behaviour according to human intuition.

4. Evaluation

4.1. Standard benchmarks

We benchmark the approach on commonly used small-scale toy scenarios for a proof of concept and a first quantitative evaluation against current state-of-the-art approaches: GPIRL (Levine et al., 2011), NPB-FIRL Choi and Kim (2013) and the original MaxEnt (Ziebart et al., 2008). Furthermore, we demonstrate that the application of convolutional layers can remove the dependency on separately defined spatial features. All experiments are performed in this scenario based on a small number of demonstration trajectories, typically less than 100. While the number of

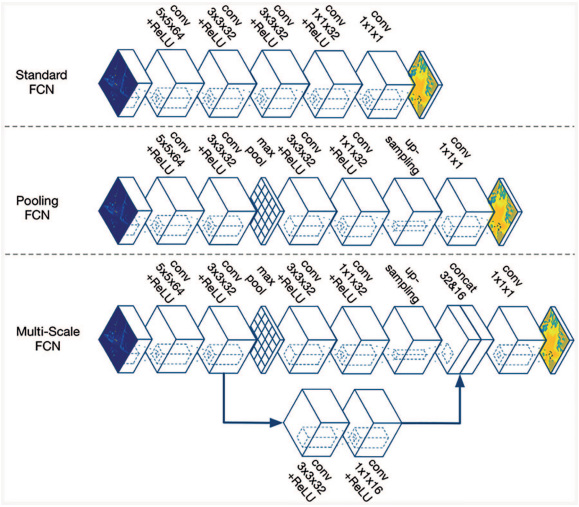


Fig. 3. Illustration of the three proposed network architectures. The standard FCN represent a serial architecture of convolutional layers. The pooling FCN adds limited translational invariance to the standard architecture. To combine the capacity for translationally variant and invariant features, the multi-scale network combines two parallel branches with and without max-pooling layer.

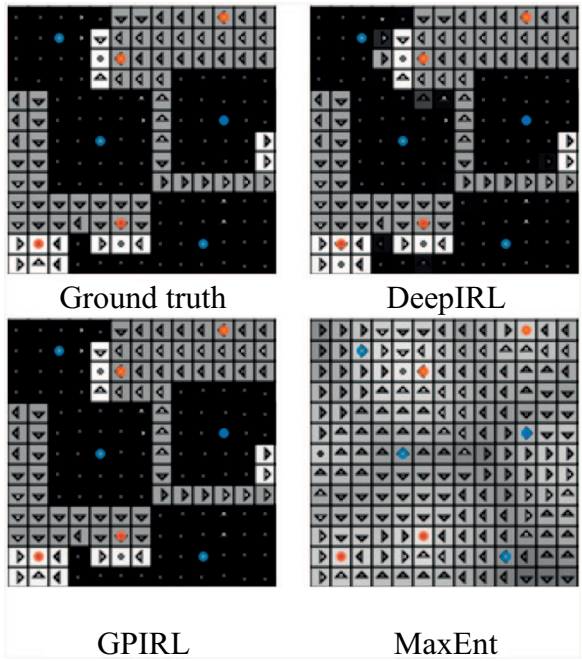


Fig. 4. Reward reconstruction sample in the Objectworld benchmark provided $N = 64$ examples and $C = 2$ colours with continuous features. White, high reward; black, low reward.

samples obviously does not represent a principal application of deep architectures, it serves to show the ability of our approach to approximate arbitrary non-linear cost functions.

As these benchmarks focus on low-complexity reward functions and very limited demonstration datasets, we employ comparably shallow networks based on two hidden layers and rectified linear units. To enable direct comparison against the other approaches, we limit the network to approximate the reward for a state only on the corresponding feature representation of that particular state, which corresponds to a FCN limited to filter widths of 1×1. This limitation is removed in Section 4.1.1 as the analysis targets the ability of the algorithm to learn spatial features and eliminate the dependency on predetermining useful representations. The implementation of the experiments is based on MatConvNet (Vedaldi and Lenc, 2014) applying AdaGrad (Duchi et al., 2011) for network training.

Since the ground truth reward is given in these scenarios, we use *expected value difference* as the principal metric of evaluation. It represents a measure for the sub-optimality of the optimal policy for the learned reward function under the true reward. The analysis focuses on evaluating each specific training scenario and a number of randomly generated test environments. The *transfer* examples serve to analyse each algorithm's ability to generalise to the true reward structure and resist over-fitting.

Both evaluation scenarios, Objectworld and Binaryworld, consist of a map of $M \times M$ states for $M = 32$, where possible actions include motions in all four directions as well as staying in place. In addition, two different sets of feature representations are used to evaluate the algorithms for discrete and continuous representations. For the continuous features $x \in \mathbb{R}^C$. each feature dimension describes the minimum distance to an object of one of C colours. Building on the continuous representation the discrete ones includes $C \times M$ binary features, where each dimension indicates whether an object of a given colour is closer than the threshold $d \in \{1, \ldots, M\}$. The ground truth reward in Objectworld depends on a two-step decision tree based on the value range of the available features, while Binaryworld is built on a more complex model which is based on counting the number of active features in the direct environment of a state. For a more detailed description of the scenarios and ground truth reward calculation, the reader is referred to Wulfmeier et al. (2015).

The algorithms are assessed with increasing number of demonstration samples on the training and test scenarios. While the original MaxEnt is inherently unable to capture the non-linear reward structure independent of the number of demonstration samples, both DeepIRL and GPIRL converge towards exact approximation as represented in Figure 4. NPB-FIRL was shown to perform on a similar level as GPIRL by Choi and Kim (2013). DeepIRL achieves performance commensurate to GPIRL with slight increase in the number of available expert demonstrations. The same behaviour is exhibited for both continuous and discrete feature representations (Figure 5).

The Binaryworld scenario differs to Objectworld in its feature representation as every state is randomly assigned

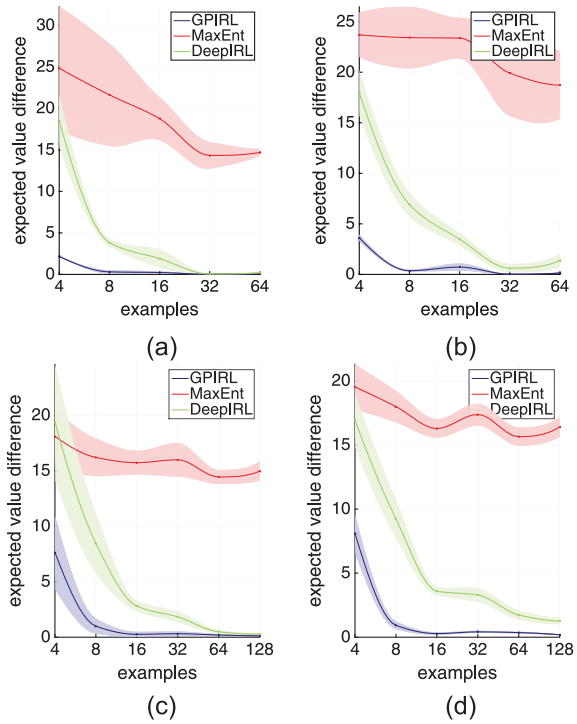


Fig. 5. Objectworld benchmark. From top left to bottom right: expected value difference (EVD) with $C = 2$ colours and varying number of demonstrations N for training (a) and transfer case (b) with continuous and respectively with discrete features in (c) and (d). As the number of demonstrations grows DeepIRL is able to quickly match performance of GPIRL on the task.

one of two colours and the feature vector for each state consequently consists of a binary vector of length 9, encoding the colour of each cell in its 3×3 neighbourhood. Since the reward depends on a more complex combination of the basic features - that is to say the number of specific features - this case is applied to represent a more complex non-linear function.

The performance of DeepIRL compared to GPIRL, linear MaxEnt and NPB-FIRL is depicted in Fig. 6. In this scenario, DeepIRL is able to learn the higher-order dependencies between features more quickly, whereas GPIRL needs a higher number of samples as the inherent kernel measure struggles to relate the reward of different examples with similarity in their state features. The NPB-FIRL algorithm struggles significantly in this test, which can be explained by the fact that describing the true reward in this scenario with the logical conjunctions is quite inefficient. In fact, it would require 2^9 different logical conjunctions, each capturing all possible combinations of features, to accurately model the reward in this framework.

One of the principal advantages of DeepIRL is its scalability with respect to the amount of training data, leading to an algorithm with constant-time execution at test time. The partial requirement of DeepIRL for more training will be rendered unimportant in robot applications based

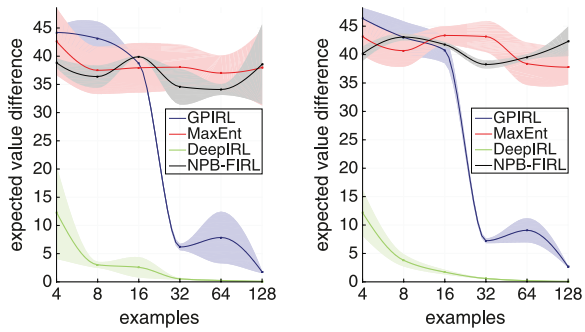


Fig. 6. Value differences observed in the Binaryworld benchmark for GPIRL, MaxEnt and DeepIRL for the training scenario (left) and the transfer task (right).

on low-cost, autonomous data acquisition, rendering representation learning and the aforementioned lower algorithmic complexity as the dominant advantage of a parametric approach.

4.1.1. Spatial Feature Learning. One principal advantage of being able to access the full gamut of different neural network architectures is the possibility to extend the model via the use of wider filters to eliminate the requirement of manual design of input features.

We demonstrate the capability of our method to learn optimal spatial features from the raw input representation. Figure 7 represents the results for both toy scenarios, but instead of using the earlier described handcrafted feature representations the input representation for our models only includes the presence of any object at every state. All spatial information has to be derived based on the convolutional filters in this context. We employed a five-layer approach with 3×3 convolutional kernels in the first two layers, followed by 1×1 width layers. By increasing the depth of the network and including wider convolutional filters, we add the capacity to the network to learn the hierarchy of features including their spatial interaction as optimised for the task at hand.

Owing to the increasing number of parameters, the approach requires additional training data to overcome overfitting and generalise similarly well. By increasing the number of demonstration samples it converges quickly towards the performance with predefined features and ultimately converges towards the exact reward function. In these simplified toy problems the earlier used handcrafted features are optimal and the true reward is directly calculated on their basis. Therefore, automatically learned features cannot exceed their performance. However, in real-world scenarios, the compression of raw data, such as images, to feature representations inherently leads to loss of information and the learning of features that optimise the utilisation of capacity with respect to the task represents a principal gain of the approach.

The test scenarios above demonstrate the feasibility of the approach on small-scale benchmarks and spatial feature

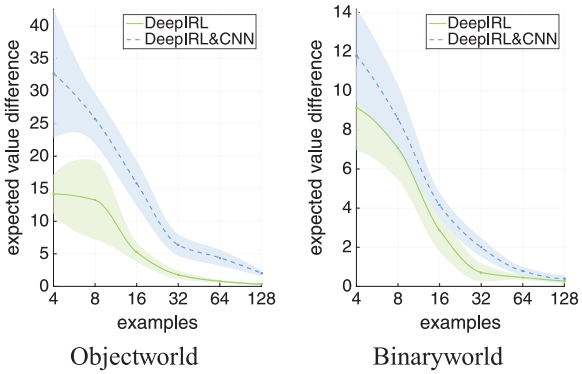


Fig. 7. Application of convolutional layers for spatial feature learning. Spatial feature learning quickly converges to performance with optimally designed features.

Fig. 8. Mobile research platform: a modified GEM golf cart.

learning with CNNs but only represent a limited evaluation. The cost functions in this context are based on handcrafted decision trees with algorithmically determined demonstration behaviours. In the following section, our method is applied in a real-world scenario to learn cost functions for motion planning for an autonomous vehicle in an urban environment.

4.2. Large-scale application

After demonstrating the capability of our model in simple, commonly used benchmarks, this section aims at evaluating its real-world applicability in the context of autonomous driving: as a regressor for cost functions used in motion planning.

4.2.1. Dataset. The driving dataset was collected over the course of 1 year involving 13 different drivers navigating pedestrian walkways and cycle lanes in the city of Milton Keynes, UK.

The data collection platform, a modified GEM (Global Electric Motorcars) golf cart (cf. Figure 8), is equipped with various sensors including 2D as well as 3D LIDARs and stereo- and mono-cameras. The relevant sensors for this work are two Velodyne HDL-32E scanners and a Bumblebee XB3 stereo-camera.

A total of over 25,000 trajectories are included for the dataset, each about 12–15 m long, from more than 120 km of urban driving. Each trajectory represents a single training input sample based on a ground-plane projection of statistics of the LIDAR data as represented in Figure 9. The trajectories were extracted with about 50–60% overlap which was empirically determined as compromise between final model performance and training time. The impact of higher overlap and therefore more samples from the same length of demonstration data on the model performance was found to be insignificant.

The network input is based on pointclouds measured by the two 3D Velodyne scanners. The pointclouds are first mapped into a grid-based static map on the ground plane of the vehicle and consecutively compressed into statistics, which include mean height, height variance and a binary indicator if the cell is visible in any scan. The grid has a size of 25 m $\times$ 25 m and a resolution of 0.25 m per cell. Based on this discretised, gridded representation, the application of FCNs to the environment representation is straightforward.

An example of the gathered demonstration trajectories is depicted in Figure 9. The trajectories are extracted from the chain of vehicle poses estimated by applying visual odometry (Churchill and Newman, 2012) on stereo images from the Bumblebee XB3 camera. The extracted transform chain is mapped into the static map frame and discretised to fit into the grid representation. The action space is simplified to a discrete set of motions around the current position. Only spatial aspects of the trajectories are being considered for this model. An expansion towards temporal aspects is discussed in Section 5.

In the following sections, we evaluate the different models against a handcrafted cost function (displayed in Figure 10). The main concept behind the design of this cost function is the detection of obstacles as areas of large height variance in each cell. Areas without LIDAR information are labelled as unknown, which in the classification task is treated as not traversable. Furthermore, obstacles are extended by the radius of the smallest circle encapsulating the GEM platform (Minkowski sum (LaValle, 2006)).

4.2.2. Prediction and classification. The model quality is evaluated based on the performance in two tasks, prediction of human behaviour and classification of traversable terrain. Taking into account that the goal of this work lies in the use as cost map for motion planning, the evaluation focuses on qualitatively and quantitatively determining the accuracy of terrain assessment.

For prediction accuracy, we apply two common metrics: the negative log-likelihood of the demonstration data (NLL) as well as the modified Hausdorff distance (MHD) (Kitani et al., 2012). The first metric represents how likely

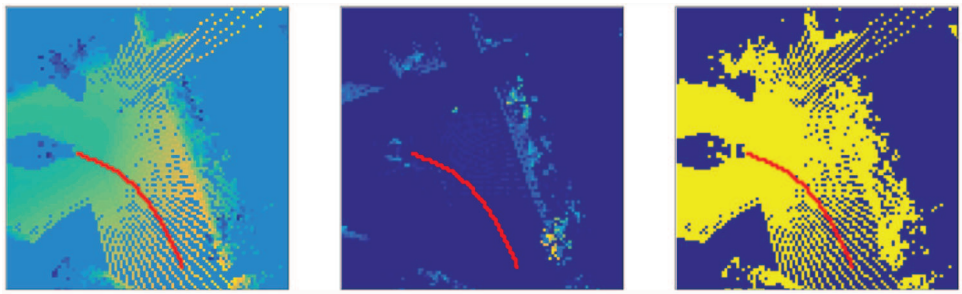


Fig. 9. Visualisation of demonstration trajectories (in red) on all channels of a static map. From left to right: mean height, height variance, cell visibility.

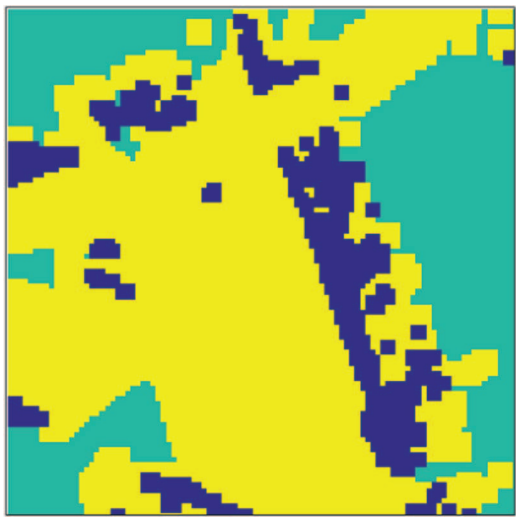


Fig. 10. Handcrafted cost function (our baseline for comparison) for the same scenario shown in Figure 9.

the expert demonstrations are given the current cost function and the latter is a spatial metric for how close the demonstrations are to samples generated from the stochastic policy in dependence of the cost map. In other words, this describes the similarity between agent and demonstration trajectories: the accuracy in imitating human driving behaviour.

The standard FCN already predicts test trajectories significantly better than the manual cost function (see Table 1) and therefore describes human driving behaviour more accurately. This result is expected as the training objective in MEDIRL (2) includes optimising the NLL of demonstration trajectories. The use of the pooling architecture without parallel information branches performs better in terms of a smooth cost function when interpreting the same areas as displayed in Figure 11. The MaxPooling approach is able to infer traversable terrain. However, the pooling step leads to a loss of spatial information as discussed in Section 3.4. Best performance is achieved when the model is given the capability to learn spatial invariance in one branch while being able to preserve feature location in a parallel chain of the MS-FCN architecture.

Table 1. Model performance on calibrated data.

Metric	NLL	MHD	FNR	FPR
Manual CF	78.13	0.284	0.441	0.000
Standard FCN	69.35	0.221	0.471	0.000
Pooling FCN	69.73	0.230	1.000	0.000
MS FCN	**65.39**	**0.200**	**0.206**	0.000

One major drawback for evaluating all approaches in a real-world driving setup is the absence of absolute ground truth for the cost map. However, the driven trajectories represent ground truth knowledge about traversable terrain and the annotation of synthetic collision trajectories is less time consuming than full manual annotation of cost maps. For this purpose, we manually annotate synthetic collision trajectories of the same number as positive samples in our classification test dataset based on pointcloud and image data, as actual collision trajectories are to be prevented. Therefore, we can overcome the impediment, evaluate our cost functions as classifiers for feasible trajectories and analyse the approaches in Type I (false positives) and Type II errors (false negatives).

In this classification setup, we need to set a threshold for deciding the minimal cost at which collisions occur and the trajectory can be classified as untraversable. This threshold is set to reduce FPR to 0%, meaning that no untraversable terrain in our test set is incorrectly classified as acceptable trajectory and planning safety is maximised.

The manual cost function has no threshold to set but is designed instead to be strictly conservative with respect to the traversability of terrain. This results in 0% false-positive rate (FPR) while showing a significant number of false negatives. While the system is tuned to produce no classification that could result in a collision, it often wrongfully rejects paths as not feasible.

Adapting the cost function thresholds to achieve 0% FPR requires elaborate tuning. While it is straightforward to achieve rates of under 5% the final adjustment to erase false positives significantly increases the false negative rate. While it leads to 47% false-negative rate (FNR) for the standard architecture, it renders the application of the pooling model infeasible as FNR of 100% represents that none of

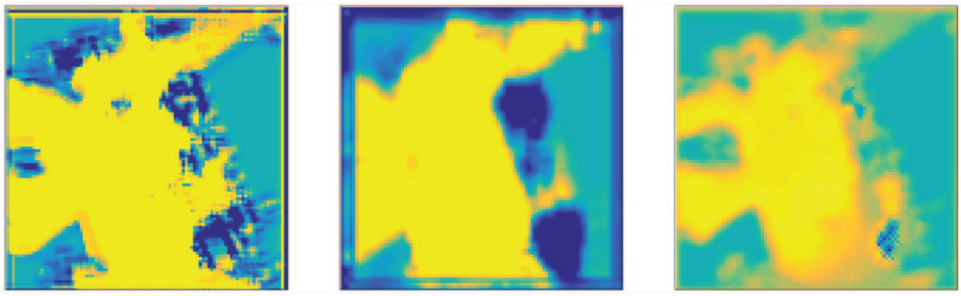


Fig. 11. Visualisation of the learned cost maps based on (a) standard FCN, (b) pooling FCN and (c) MS FCN.

the paths are classified as traversable. The MS FCN, being able to combine the benefits of spatially variant and invariant features (see Section 3.4), performs best with a reduction in FNR of about 50% in comparison with the manual cost map.

4.2.3. Robustness to systematic noise. A principal gain in learning the cost function from raw features is the significant increase in robustness towards systematic flaws in the configuration of the robot. An inaccurate calibration for example will lead to complete failure for a manually crafted cost function. As presented in the example in Figure 12, it can result in artificial obstacles due to an imprecise calibration of the pitch angle between the platform and one of the LIDARs of as little as 1°. Owing to the introduced perturbation in the pitch angle of the right Velodyne, the manually defined cost map creates obstacles in this instance. This is caused by the increased height variance of points in a specific cell which now directly depends on a cell's distance from the vehicle. The learned cost map is able to handle the bias and infer realistic cost maps even in the presence of miscalibration. As long as the input representation is rich enough to theoretically distinguish between the new features, describing artificial obstacles in the manual cost map, and real walls, etc. the system learns to approximate the decision boundary that separates traversable from untraversable terrain.

By applying the metrics introduced in Section 4.2.2, we evaluate the performance of the MS FCN architecture versus the manually defined cost function. The evaluation shown in Table 2 emphasises that the handcrafted cost function leads to nearly impassable cost maps in this scenario with an FNR of more than 97%. The learned model, on the other hand, gives an FNR that, although comparatively worse than the case with the correct calibration (cf. Table 1), still remains within functional range.

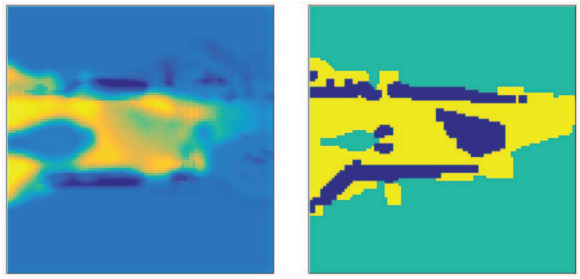


Fig. 12. Example cost maps based on miscalibrated data with MS FCN (left), and the handcrafted cost function (right).

Table 2. Model performance on miscalibrated data. By applying the MS FCN architecture we gain significantly more robustness towards systematic bias in the sensor calibration in comparison with a handcrafted cost function.

Metric	NLL	MHD	FNR	FPR
Manual CF	89.40	0.432	0.971	0.000
MS FCN	**69.35**	**0.267**	**0.525**	0.000

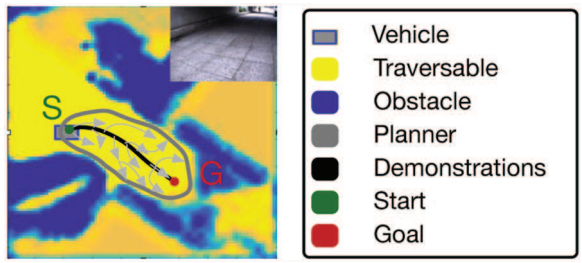


Fig. 13. Illustration of sparse feedback, showing a demonstration trajectory on the spatial cost map around the vehicle, as well as the region explored by the planning algorithm. Error feedback is only created for the area surrounding sample trajectories.

4.3. Addressing real-world challenges

When transferring the MEDIRL approach from toy scenarios to full-scale, urban driving scenarios, additional challenges emerge. The iterative nature of planning and refining the cost model in the training process, as described in Section 3.3, leads to a principal focus on learning discriminative features around highly visited states as depicted in Figure 13. Features for terrains that are neither explored by demonstration samples nor the planning step will not be formed by backpropagation of error terms through the network. The model has to generalise to describe these states based on similarity to more commonly traversed areas, resulting in inaccuracies, artefacts and noisy reward maps.

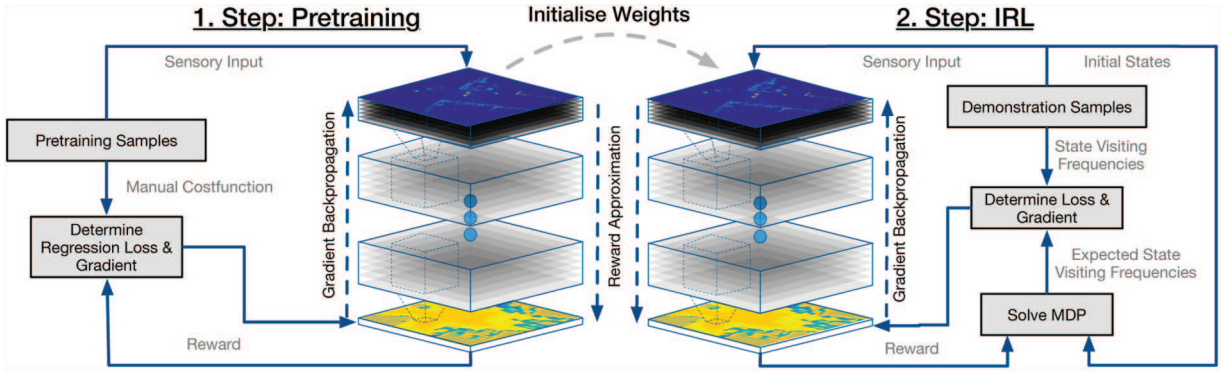


Fig. 14. Schema for additional network pretraining, where the model learns to regress to a manual prior cost map. Subsequently the network is fine-tuned to predict the reward under the MEDIRL framework.

While this does not represent any limitations for the toy scenarios, it is of concern where feature representations for similar places differ based on their position in state space, as is the case in the dataset used in this work. The LIDAR scan points in this setup will be spread sparser at greater distance from the car, resulting in spatially variant representations for the same objects in our environment. This behaviour is illustrated in the cost maps in Figure 11 where areas with sparser LIDAR which differ more significantly from the traversed areas display more grainy and noisy cost representations.

To illustrate the problem further, we consider the following example. When training a model for image segmentation, the objective creates feedback for each individual pixel. When training with a loss based on IRL on the other hand, error terms will focus on the region around the demonstration trajectories. These error terms are based on states visited by the demonstration samples and the planning algorithm, which inherently focuses around sample data. Our work addresses these shortcomings by pretraining the network towards a dense human-provided prior, as visualised in Figure 14, to learn richer feature representations for untraversed areas and increase the network's ability to generalise.

These cost maps can be automatically generated from the laser input data based on manually handcrafted features, which enables us to utilise the availability of large amounts of data without human labelling efforts. However, the principal benefit is the ability to explore all features relevant to generating the cost function, leading to better generalisation in areas with greater distance from the demonstration trajectories.

To increase the performance of our approaches as path planning cost maps, we add more common preprocessing steps to the dataset from Section 4.2, such as normalising the input data, and training on shorter trajectories that are more representative for the motion primitives employed in the final motion planning framework of closer to 10 m. For the normalisation procedure we center each feature with zero mean and unit standard variation. When performing path planning, a threshold has to be determined to define untraversable and unsafe terrain. To simplify this step we normalise the output to range [0, 1] by applying a final sigmoid activation function. Furthermore, we introduce early stopping based on validation performance to increase generalisation and prevent overfitting as well as the complete overriding of prior information from the weight initialisation. For this procedure we randomly separate 5% of our training data as a validation set.

Table 3. Evaluation of cost functions for urban driving under the negative log-likelihood (NLL) and modified Hausdorff distance (MHD) metrics. Lower numbers represent models that are approximating human behaviour with higher precision.

Metric	NLL	MHD
Manual cost function	56.402	0.286
w/o pretraining	47.535	0.218
w pretraining	**46.767**	**0.182**

Utilising prior knowledge in this approach improves accuracy for prediction and therefore imitation of human driving behaviour as displayed in Table 3. However, its principal gain lies in being able to improve the robustness and spatial generalisation of the learned cost functions, leading to more accurate classification of traversable terrain. The precision–recall curves for networks with and without pretraining are depicted in Figure 15. A single point represents the manual cost function as it does not include a threshold parameter. This approach is manually designed to be conservative and enables us to operate at very good precision. However, it falsely rejects much of the terrain incorrectly as untraversable as described in Section 4.2.2. In contrast, the learned cost functions find possible paths in many situations where the manual cost function will get stuck. The additional introduction of prior knowledge into the training process achieves a significant gain in precision compared with random initialisation. Hence, utilising this knowledge is an important step towards robust application of the learned cost maps.

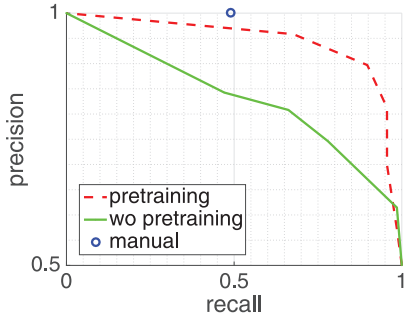


Fig. 15. Precision–recall (PR) curves for trajectory classification. The manual cost function has high precision but low recall, meaning that it is safe but conservative and will falsely classify a significant number of feasible trajectories as untraversable. Applying human priors in the pretraining step enables a significant gain in precision towards the baseline. This method approaches the precision of the manual cost function while strongly exceeding it in recall.

When explicitly handcrafting a cost function, corner-cases, as displayed in Table 4, can represent relevant shortcomings. The handcrafted cost function as described in Section 4.2 can lead to inaccuracies in the presence of slopes, which can exceed the threshold and will be shown as untraversable. Stairs, on the other hand, can still fit within the same threshold, but present obstacles for any wheeled vehicle as they cannot be traversed due to their discontinuity. Furthermore, underpasses might incorrectly exceed the height threshold for obstacles since scans from ceiling and floor result in a high height range. Bollards that are extended slightly too far will seem untraversable and areas such as grass can look very similar in features to pathways but should not be traversed.

The randomly initialised network already learns to represent the main obstacles and traversable areas, but it results in some noisy areas and artificial obstacles. When initialised based on prior domain knowledge, the network additionally learns to refine the representation and is significantly more robust. It learns to represent distinct obstacle boundaries and displays fewer artefacts. This approach is able to distinguish slopes from stairs and extends obstacle boundaries more precisely as necessary for safe traversal as seen in the respective cases in Table 4.

As additional safety benefit, we can define how far we trust our perception systems by adapting the length of demonstration trajectories. The learned cost functions show untraversable terrain starting at about 13 m distance from the vehicle position, which is the length of the demonstration trajectories. Since features in distant areas are only traversed by the planning step and not demonstration samples in the training process, they will be classified as untraversable with high probability.

We argue that without pretraining, the expressive power of the network is employed to learn the very specific representation focused around the demonstration trajectories as this strongly influences the training objective. When we instead pretrain the model to predict an existing cost map and learn with more dense feedback for the whole environment, the remainder of the training process is able to better capture some of the corner cases described previously. The approach results in more distinct obstacle boundaries and more robust cost maps as demonstrated in this section.

5. Discussion

We argue for the use of a high-capacity, parametric approach to IRL using FCNs, in order to tractably approximate the cost function in a complex real-world scenario with maximum accuracy and constant-time operation.

In order to achieve a task of this complexity with manually handcrafted features, one requires significant expert knowledge of the application domain, sensory pipeline and decision-making approach. Further, as demonstrated in Section 4.2, some necessary features can be easily missed in the preprocessing setup given the amount of variation in domain-specific obstacles. In contrast, the spatial features as represented by the filters of the FCN on the other hand are inherently optimised for this task.

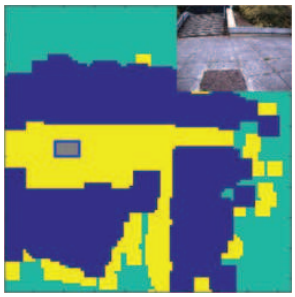
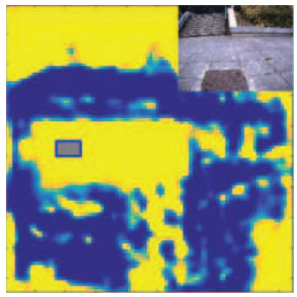
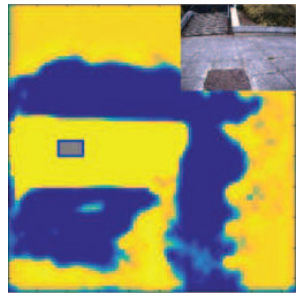
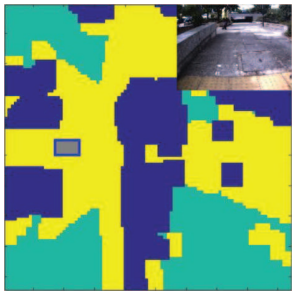
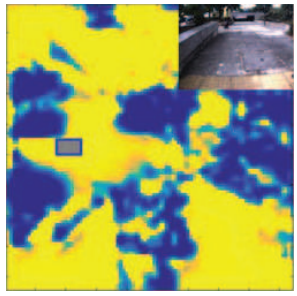
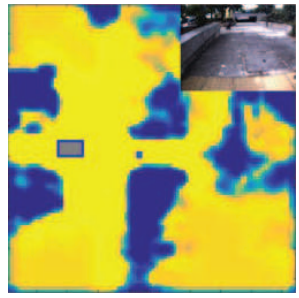
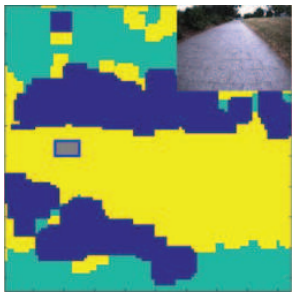
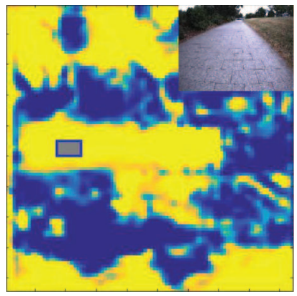
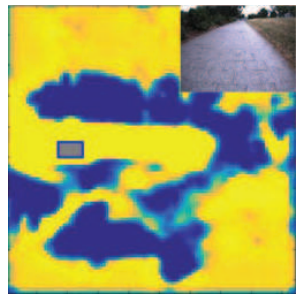
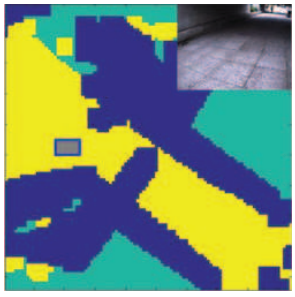
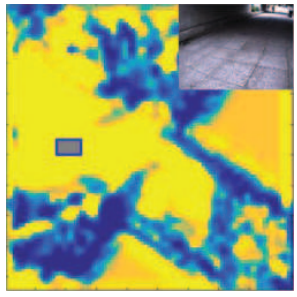
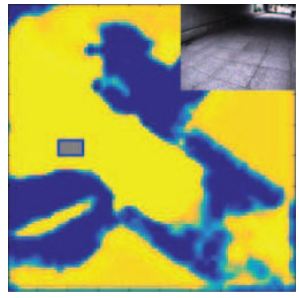
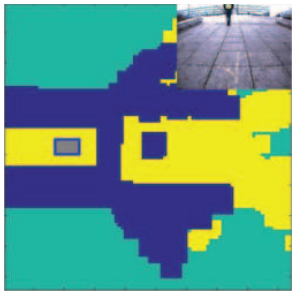
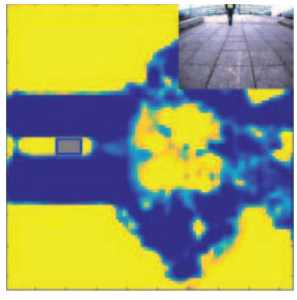
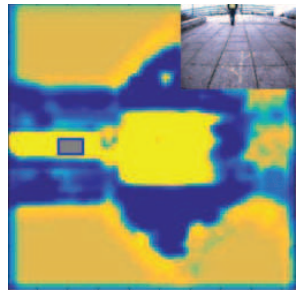
While non-parametric approaches such as GPIRL possess the capacity to learn complex non-linear cost functions when given these hypothetically perfect features, but do not exhibit constant-time behaviour and are rendered intractable with the large amounts of required training data.

Additional benefits of the proposed approach include robustness towards unknown, systematic noise, as exemplified by miscalibration in Section 4.2.3. This capability does not exist for approaches that rely on predetermined feature representations, because this would require knowledge of the applied noise when constructing features for the task.

As is common for systems that can affect their environment, the vehicle can encounter a significantly different sensor data distribution during test time. While IRL already generalises better than direct imitation learning in this case (Argall et al., 2009), the problem can be additionally addressed by adapting a DAgger-like (Ross et al., 2010) approach, where additional demonstration samples are collected for environments that the system encounters in closed loop.

While our model learns the spatial preferences underlying human driving behaviour, the approach, in its current form, does not address the velocity profiles alongside driven paths as well as aspects of temporal consistency between consecutive cost maps. This extension requires the processing of sensor inputs across the extend trajectory instead of focusing on a limited period around the start of each trajectory, since the position of dynamic obstacles will change during the traversal of a given trajectory. Both mentioned challenges can be partially addressed on the model side with extension towards recurrent network models as well as temporal convolutions or as separate post-processing step. Large-scale data collection efforts as currently predominant in the industrial sector can build the foundation for training these more complex, high-capacity models to

Table 4. Corner cases for the cost function. The images include views from the front facing camera module with the vehicle represented as a grey rectangle driving towards the right side of each cost map. Obstacles are represented in blue, while yellow depicts traversable terrain.

Scenario	Manual cost function	w/o pretraining	w pretraining
Stairs			
Bollards			
Grass			
Underpass			
Slope			

expose the factors that determine temporal aspects of taken trajectories.

6. Conclusions and future directions

We have developed a highly scalable approach to learning cost functions for autonomous mobile platforms from large collections of demonstration samples for human driving. Common simulation benchmarks have been employed to prove the efficacy of the algorithm and confirm that it can approximate the given ground truth reward functions, with convergence commensurate to current state-of-the-art IRL approaches. We have demonstrated the principal benefits of MEDIRL by applying it to an urban driving scenario based on over 120 km of driving with over 25,000 demonstration samples. The scale and complexity exceeds the reach of existing approaches, either due to computational intractability (such as for GPIRL) or limited representative capacity (such as for linear IRL).

By employing an end-to-end method that maps directly from low-level features to cost maps, the approach is able to correct for systematic biases and learn task-optimal features which help our method to perform well even in the presence of systematic noise as demonstrated in Section 4.2.3 by applying it to a miscalibrated dataset.

Additional challenges emerging from the application on real-world sensor data in the large-scale evaluation such as spatially sparse feedback are furthermore addressed by incorporating human priors into a network pretraining step. This allows the MEDIRL framework to perform parameter fine-tuning based on an educated human prior cost map.

Additional benefits might be found in a semi-supervised training procedure, exploiting both reconstruction and IRL-based cost functions for the final training procedure as well as utilising additional input representation samples in the absence of driven trajectories. The approach can help to address overfitting and increase generalisation performance.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported by the UK's Engineering and Physical Sciences Research Council (EPSRC) through the Doctoral Training Award (DTA) and under grants EP/J012017/1 and EP/M019918/1 as well as the Hans-Lenze-Foundation.

References

Abbeel P, Dolgov D, Ng A and Thrun S (2008) Apprenticeship learning for motion planning with application to parking lot navigation. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008. IROS 2008*, pp. 1083–1090.

Abbeel P and Ng AY (2004) Apprenticeship learning via inverse reinforcement learning. In: *Proceedings of the twenty-first international conference on Machine learning*. New York: ACM Press.

Argall BD, Chernova S, Veloso M and Browning B (2009) A survey of robot learning from demonstration. *Robotics and Autonomous Systems* 57(5): 469–483.

Chen C, Seff A, Kornhauser A and Xiao J (2015) Deepdriving: Learning affordance for direct perception in autonomous driving. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2722–2730.

Choi J and Kim KE (2013) Bayesian nonparametric feature construction for inverse reinforcement learning. In: *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*. AAAI Press, pp. 1287–1293.

Choset HM (2005) *Principles of robot motion: theory, algorithms, and implementation*. Cambridge, MA: MIT Press.

Churchill W and Newman P (2012) Continually improving large scale long term visual navigation of a vehicle in dynamic urban environments. In: *15th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE, pp. 1371–1376.

Duchi J, Hazan E and Singer Y (2011) Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research* 12: 2121–2159.

Finn C, Levine S and Abbeel P (2016) Guided cost learning: Deep inverse optimal control via policy optimization. *CoRR* abs/1603.00448.

Hinton GE, Srivastava N, Krizhevsky A, Sutskever I and Salakhutdinov R (2012) Improving neural networks by preventing co-adaptation of feature detectors. *CoRR* abs/1207.0580.

Kitani KM, Ziebart BD, Bagnell JA and Hebert M (2012) Activity Forecasting. In: Fitzgibbon A, Lazebnik S, Perona P, Sato Y and Schmid C (eds.) *Computer Vision - ECCV 2012* (Lecture Notes in Computer Science, vol. 7575). Berlin: Springer, pp. 201–214.

Kretzschmar H, Spies M, Sprunk C and Burgard W (2016) Socially compliant mobile robot navigation via inverse reinforcement learning. *The International Journal of Robotics Research*, in press.

Kuderer M, Gulati S and Burgard W (2015) Learning driving styles for autonomous vehicles from demonstration. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 2641–2646.

LaValle SM (2006) *Planning algorithms*. Cambridge: Cambridge University Press.

Levine S, Popovic Z and Koltun V (2010) Feature construction for inverse reinforcement learning. In: *Advances in Neural Information Processing Systems*, pp. 1342–1350.

Levine S, Popovic Z and Koltun V (2011) Nonlinear inverse reinforcement learning with gaussian processes. In: *Advances in Neural Information Processing Systems*, pp. 19–27.

Liu F, Shen C, Lin G and Reid ID (2015) Learning depth from single monocular images using deep convolutional neural fields. *CoRR* abs/1502.07411.

Long J, Shelhamer E and Darrell T (2015) Fully convolutional networks for semantic segmentation. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431–3440.

Lopes M, Melo F and Montesano L (2009) Active learning for reward estimation in inverse reinforcement learning. In: *Machine Learning and Knowledge Discovery in Databases*. New York: Springer, pp. 31–46.

Montemerlo M, Becker J, Bhat S, et al. (2008) Junior: The stanford entry in the urban challenge. *Journal of Field Robotics* 25(9): 569–597.

Nguyen QP, Low BKH and Jaillet P (2015) Inverse reinforcement learning with locally consistent reward functions. In: Cortes C, Lawrence ND, Lee DD, Sugiyama M and Garnett R (eds.) *Advances in Neural Information Processing Systems 28*. Curran Associates, Inc., pp. 1738–1746.

Pfeiffer M, Schwesinger U, Sommer H, Galceran E and Siegwart R (2016) Predicting actions to act predictably: cooperative partial motion planning with maximum entropy models . In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Pomerleau DA (1990) Neural network based autonomous navigation. In: *Vision and Navigation*. New York: Springer, pp. 83–93.

Ramachandran D and Amir E (2007) Bayesian inverse reinforcement learning. *Urbana* 51: 61801.

Ratliff ND, Bagnell JA and Zinkevich MA (2006) Maximum margin planning. In: *Proceedings of the 23rd International Conference on Machine Learning*. New York: ACM Press, pp. 729–736.

Ratliff ND, Silver D and Bagnell JA (2009) Learning to search: Functional gradient techniques for imitation learning. *Autonomous Robots* 27(1): 25–53.

Ross S, Gordon GJ and Bagnell JA (2010) A reduction of imitation learning and structured prediction to no-regret online learning. In: *Proceedings of the International Conference on Artifical Intelligence and Statistics*.

Sermanet P, Eigen D, Zhang X, Mathieu M, Fergus R and LeCun Y (2013) Overfeat: Integrated recognition, localization and detection using convolutional networks. *CoRR* abs/1312.6229.

Sermanet P, Scoffier RHM, Grimes M, et al. (2010) A multirange architecture for collision-free off-road robot navigation. *Journal of Field Robotics* 26: 52–87.

Szegedy C, Liu W, Jia Y, et al. (2015) Going deeper with convolutions. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Thrun S, Montemerlo M, Dahlkamp H, et al. (2006) Stanley: The robot that won the darpa grand challenge. *Journal of Field Robotics* 23(9): 661–692.

Urmson C, Anhalt J, Bagnell D, et al. (2009) *Autonomous Driving in Urban Environments: Boss and the Urban Challenge*. Berlin: Springer, pp. 1–59.

Vedaldi A and Lenc K (2014) Matconvnet – convolutional neural networks for matlab. *CoRR* abs/1412.4564.

Wulfmeier M, Wang DZ and Posner I (2016a) Watch this: scalable cost-function learning for path planning in urban environments. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Wulfmeier M, Rao D and Posner I (2016b) Incorporating human domain knowledge into large scale cost function learning. In: *Neural Information Processing Systems Conference, Deep Reinforcement Learning Workshop*, volume abs/1612.04318.

Wulfmeier M, Ondruska P and Posner I (2015) Maximum entropy deep inverse reinforcement learning. In: *Neural Information Processing Systems Conference, Deep Reinforcement Learning Workshop*, volume abs/1507.04888.

Zhifei S and Joo EM (2012) A review of inverse reinforcement learning theory and recent advances. In: *2012 IEEE Congress on Evolutionary Computation*. IEEE, pp. 1–8.

Ziebart BD, Maas AL, Bagnell JA and Dey AK (2008) Maximum entropy inverse reinforcement learning. In: *AAAI*, pp. 1433–1438.