2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Daejeon Convention Center
October 9-14, 2016, Daejeon, Korea

Watch This: Scalable Cost-Function Learning for Path Planning in Urban Environments

Markus Wulfmeier[1], Dominic Zeng Wang[1] and Ingmar Posner[1]

***Abstract*— In this work, we present an approach to learn cost maps for driving in complex urban environments from a large number of demonstrations of human driving behaviour. The learned cost maps are constructed directly from raw sensor measurements, bypassing the effort of manually designing cost maps as well as features. When deploying the cost maps, the trajectories generated not only replicate human-like driving behaviour but are also demonstrably robust against systematic errors in putative robot configuration. To achieve this we deploy a Maximum Entropy based, non-linear IRL framework which uses Fully Convolutional Neural Networks (FCNs) to represent the cost model underlying expert driving behaviour. Using a deep, parametric approach enables us to scale efficiently to large datasets and complex behaviours while being run-time independent of dataset extent during deployment. We demonstrate scalability and performance on an ambitious dataset collected over the course of one year including more than 25k demonstration trajectories extracted from over 120km of driving and 13 different drivers. We evaluate against a carefully designed cost map and, in addition, demonstrate robustness to systematic errors by learning precise cost-maps even in the presence of system calibration perturbations.**

I. Introduction

The majority of state-of-the-art motion planning systems for autonomous driving applications rely on manually designed cost-functions, with recent successful examples given by the competing teams in the DARPA Grand and Urban Challenges [1], [2]. When designing a cost-function, obstacles typically are inflated as a function of the vehicle size. The weighting of costs from different sensing modalities relies on extremely detailed domain knowledge. In addition, designing good features to extract from raw input data for computing the cost maps is often a non-trivial task relying heavily on a well-understood hardware setup.

The requirement for high-capacity models for cost-functions arises when one considers the application of planning frameworks in urban environments that are of significant complexity. For example, consider a light-weight electric vehicle designed to transport people in a city between popular locations such as the train station and shopping centres. This scenario introduces new challenges to the planning framework. In addition to coping with conventional obstacles in usual urban driving scenarios, such as trees and cars, the planner is faced with additional, unconventional obstacles, such as bollards, narrow underpasses and steep ramps that are easily navigated by pedestrians, but challenging for a robot.

[1]The authors are with the Mobile Robotics Group, Department of Engineering Science, University of Oxford, United Kingdom;
markus, dominic, ingmar@robots.ox.ac.uk

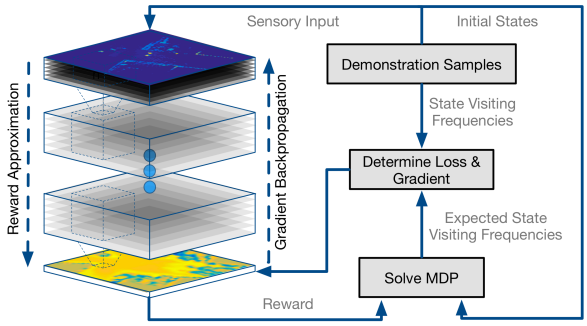


Fig. 1: Schema for training neural networks in the Maximum Entropy paradigm for IRL.

Due to the described requirements and problems in manual cost function design, our approach focuses on learning end-to-end mappings from sensory perception to cost based on large amounts of expert demonstrations. Furthermore, this approach provides additional robustness towards systematic inaccuracies that can be found e.g. in system calibration, consequently rendering it more independent of exact knowledge of vehicle configuration.

In this work, we formulate cost-function learning from expert demonstrations as an Inverse Reinforcement Learning (IRL) problem [3]. Recently, Wulfmeier et al. [4] proposed a framework that introduced training of deep neural networks into the paradigm of Maximum Entropy IRL [5]. The method is based on iterative refinement of the cost-model interwoven with the solution of the planning problem formulated as a Markov Decision Process (see Figure 1 for an illustration of the process). Representing the cost-function with a deep architecture is attractive because it opens up the possibility of learning high-capacity, highly non-linear models that are necessary for describing complex, real-world urban environments. However, while the original proof-of-concept work targeted the feasibility as well as performance on toy scenarios, the proposed network architecture as it stands does not have enough capacity for our application at hand. In this work, we scale up the framework proposed in [4] to cope with the full complexity of real-world urban driving and produce increased sample efficiency through convolutional layers. There are several advantages of this approach. First, by learning from human drivers, we learn cost-functions that generate trajectories that mimic human driving, closer to what a passenger will expect. Second, as the proper behaviours at certain types of environments are shown by human experts, we may learn to automatically correct for any systematic biases (such as a misaligned calibration),

or traverse an otherwise untraversable path (e.g. through a tight pair of bollards, where a conventional planner will have no choice but be conservative). Lastly, deploying a deep convolutional neural network enables us to learn good feature representations directly from raw sensor input.

The key contributions of this paper are three-fold:

1) New architectures for the Maximum Entropy Deep IRL framework [4] that enable it to learn high-capacity, non-linear models that are necessary for handling the complex environments encountered in real-world urban driving scenarios.
2) Evaluation of the approach on a dataset capturing the natural driving behaviours of 13 different human drivers, resulting in over 25,000 training samples.
3) A demonstration of the robustness of the proposed approach to systematic biases in robot configuration, such as sensor miscalibration.

To show the efficacy of the proposed approach, we also compare it against a carefully handcrafted cost-function for path planning that is currently deployed on our research platforms. As the evaluation on our year-long dataset in Section IV shows, the proposed approach outperforms significantly the hand-designed cost-function. In addition we show that, even under situations of systematic biases such as sensor miscalibration, the proposed approach remains functional, while the hand-designed cost-function fails catastrophically.

II. Related Works

Recent works in applying learning from demonstration (LfD) to robotic tasks have led to great advances in various approaches including direct Policy Imitation and IRL [6]. A main distinction in LfD is found in the type of model that is derived, with Policy Imitation focusing on directly learning an agent's policy and IRL in contrast targeting to infer the agent's underlying reward structure.

The reward model is generally seen to be more succinct than the policy and conceived to be preferable as generalisation becomes important [3]. While direct policy imitation enables the design of reactive controllers, long-term decision making systems aiming to find a safe, collision-free path in cluttered terrains need to plan further ahead and test plans against constraints. An additional benefit of model-based IRL approaches is their increased sample efficiency.

Due to its strengths for planning tasks and the capability for integration into existing systems, recent work on IRL lead to significant success for long-term planning tasks. Results include improving driving and robotic navigation with focus on interaction of mobile autonomous platforms and humans [7]. In contrast to these works, we develop an approach that scales with both the size of the training set and additional complexity in the input data by applying Fully Convolutional Neural Networks (FCNs) [8] with the principal goal of a direct perception-to-cost mapping and robustness towards systematic biases in the configuration of the robot.

Maximum Entropy IRL [5] represents a state-of-the-art approach to learning reward mappings by framing the demonstration trajectories as drawn from a distribution with the probability for a trajectory only depending on the expected future rewards. While the method addresses expert suboptimality in an efficient way it only provides a linear reward model that can easily be overburdened with approximating complex reward functions. In parallel to this work, Finn et al developed a sampling based approach for IRL to train non-convolutional neural networks in a MaxEnt framework for tasks of robotic manipulation and navigation [9].

Non-parametric methods such as Gaussian Processes (GPs) have been employed to overcome these limitations [10] and while this in principle extends the IRL paradigm to the flexibility of nonlinear reward approximation, the use of a nonparametric model makes it prone to requiring a large number of demonstration samples in order to approximate highly varying reward functions [11]. The task addressed in this work leads to complex reward mappings based directly on sensory data without manual feature design and therefore high demands regarding to the dataset size which quickly renders any nonparametric approach computationally impracticable.

Our work addresses complex, non-linear reward functions by applying FCNs to learn necessary feature representations and reward functions for IRL similar to [4] and extends towards the application in a large scale robotics scenario with tens of thousands of demonstration trajectories.

III. Methodology

We address the task of learning a cost-function for the motion planning system by applying deep learning to the Maximum Entropy paradigm for IRL. The resulting training algorithm focuses on the ideas of dynamic programming and backpropagation. Compared to early work [4], we employ convolutional networks and extend their architecture to handle features at multiple scales using parallel information streams based on a directed acyclic graph.

A. Inverse Reinforcement Learning

The principal goal of IRL is to infer the preferences underlying specific behaviours. The task is often solved based on framing the task in the Markov Decision Process (MDP) framework, which can be defined as $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma, r\}$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ denotes the set of possible actions, $\mathcal{T}$ denotes the transition model, γ is the discount factor - a value in $(0, 1]$ reducing the influence of future rewards - and finally r is the reward structure. The terms "cost" and "reward" are used interchangeably in the following sections since one can be converted into the other by negation.

IRL specifically addresses the case where, instead of the reward structure, a set of expert demonstrations $\mathcal{D} = \{\varsigma_1, \varsigma_2, ..., \varsigma_N\}$ are supplied. Each demonstration consists of a sequence of state-action pairs such that $\varsigma_i = \{(s_1, a_1), (s_2, a_2), ..., (s_K, a_K)\}$. The principal goal is to infer the underlying reward r of the expert demonstrations, which can be used either to predict behaviour or even to replicate it.

The general formulation of IRL results in two main complications. The expert has to be modelled in a way that suboptimality with respect to the reward can be handled. Furthermore, additional constraints have to be introduced to dissolve the inherent ambiguity of reward structures for given demonstrator behaviour [3]. For this work, the robust handling of suboptimality is of absolute necessity since the demonstration trajectories are extracted from large driving datasets without specified driving behaviour and multiple drivers as further explained in section IV.

B. Maximum Entropy Deep Inverse Reinforcement Learning

The Maximum Entropy paradigm (MaxEnt) for IRL addresses the suboptimality and reward ambiguity problems mentioned in section III-A by modelling expert behaviour as a distribution over expert trajectories and constraining this distribution to the one of highest entropy [5]. Furthermore, as shown in [4], it leads to a fully differentiable objective function, thus enabling backpropagation of loss gradients and lends itself naturally for training neural network architectures. We formulate the training procedure in what follows as a straightforward stochastic gradient descent optimisation.

The MaxEnt formulation defines the agent's policy $\pi_D(a|s)$ such that the resulting behaviour is directed towards the maximisation of rewards given the current model. This leads to the probability for user preference of any given trajectory between specified start and goal states being proportional to the exponential of the reward along the path:

$$P(\varsigma|r) = \prod_{i=1}^{K} \pi_D(a_i|s_i) \propto \exp\{\sum_{i=1}^{K} r_{s_i,a_i}\}. \quad (1)$$

The complete objective for Maximum Entropy Deep Inverse Reinforcement Learning [4] is based on a data term, maximising the likelihood of demonstration data given the parametrised reward function, as well as a model term for regularisation purposes:

$$\mathcal{L}(\theta) = \log P(\mathcal{D}, \theta|r(\theta)) = \underbrace{\log P(\mathcal{D}|r(\theta))}_{\mathcal{L}_\mathcal{D}} + \underbrace{\log P(\theta)}_{\mathcal{L}_\theta}. \quad (2)$$

Examples of useful regularisers include the l_1 and l_2 norms or a combination of both (Elastic Net).

The data-based gradient given by the Maximum Entropy approach can be separated into two gradients by the application of the chain rule – the gradient of the objective with respect to the reward and the gradient of the reward with respect to the network parameters [4]:

$$\frac{\partial \mathcal{L}_\mathcal{D}}{\partial \theta} = \frac{\partial \mathcal{L}_\mathcal{D}}{\partial r}\frac{\partial r}{\partial \theta} \quad (3)$$

$$= \underbrace{(\mu_\mathcal{D} - \mathbb{E}[\mu])}_{\text{MaxEnt}} \underbrace{\frac{\partial}{\partial \theta} r(\theta)}_{\text{Backpropagation}}. \quad (4)$$

By extending the original linear formulation to neural networks, we benefit from the efficiency of gradient backpropagation to associate the difference in state visitation frequencies μ with the most relevant parameters.

Algorithm 1 shows the procedure for iterative refinement of the reward model based on gradients computed with Equation 3, while Figure 1 intuitively visualises the training procedure. The expert's state visitation frequencies μ_D represent the number of visits to a specific state extracted from the training data and represents the sum over actions for the state-action visitation frequencies μ_D^a. The learner's expected state visitation frequencies $\mathbb{E}[\mu]$ are being calculated following Algorithm 1. The input to the neural network (the sensor measurement) is represented by f. A more detailed illustration of the algorithm including the dynamic programming formulation for value iteration and policy propagation in lines 4 & 5 can be found in [4].

Algorithm 1 Maximum Entropy Deep IRL

Input: $\mu_D^a, f, S, A, T, \gamma$
Output: optimal weights θ^*
1: $\theta^1 = \text{initialise\_weights}()$
Iterative model refinement
2: **for** n = 1 : N **do**
3: $r^n = \text{nn\_forward}(f, \theta^n)$
Solution of MDP with current reward
4: $\pi^n = \text{approx\_value\_iteration}(r^n, S, A, T, \gamma)$
5: $\mathbb{E}[\mu^n] = \text{propagate\_policy}(\pi^n, S, A, T)$
Maximum Entropy loss and gradients
6: $\mathcal{L}_D^n = \log(\pi^n) \times \mu_D^a$
7: $\frac{\partial \mathcal{L}_D^n}{\partial r^n} = \mu_D - \mathbb{E}[\mu^n]$
Compute network gradients
8: $\frac{\partial \mathcal{L}_D^n}{\partial \theta_D^n} = \text{nn\_backprop}(f, \theta^n, \frac{\partial \mathcal{L}_D^n}{\partial r^n})$
9: $\theta^{n+1} = \text{update\_weights}(\theta^n, \frac{\partial \mathcal{L}_D^n}{\partial \theta_D^n})$
10: **end for**

C. Multi-Scale Architectures

In addition to the basic Fully Convolutional Network (FCN) presented in [4] a main contribution of this work is the design of two extended architectures based on advances in other applications for neural networks to address shortcomings of the original network.

The pooling FCN illustrated in Figure 2 simply introduces a notion of translation invariance and reduces the size of the representation in the following layers. The extension is needed to address the influence of features on the following layers with limited spatial invariance. Especially in the application in section IV, the areas further away from the car cannot be densely sensed via LIDAR and the resulting patterns in all feature channels can be hard to model in all possible variations. While a larger number of low level filters might be able to address the issue, it will increase the chance of overfitting. Max-Pooling layers introduce a limited invariance with respect to translations which reduces the modelling efforts and increases accuracy without increasing the chance of overfitting.

While the pooling based architecture reduces the spatial size of the representation and leads to increased spatial

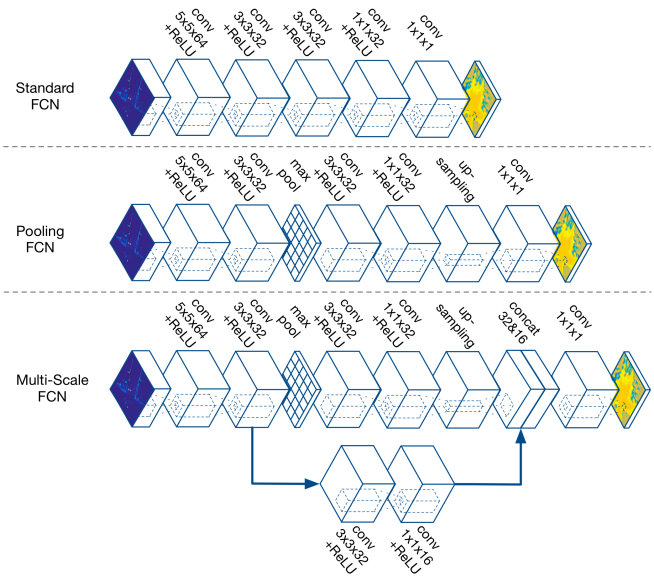


Fig. 2: Illustration of the three proposed network architectures.

invariance, this also leads to the loss of location information for low level features. Inspired by work on the multi-scale Deep Jet architecture [8] that integrates features of different scales, we introduce the Multi-Scale (MS) FCN architecture shown in Figure 2. The difference between our proposed MS FCN architecture and the Deep Jet architecture is that, instead of summing the contributions from the two branches of the network, we concatenate them following [12]. This modification to the Deep Jet architecture is important because in our case the semantic meanings of the feature channels from the two branches differ. Hence to prevent mixing of the two channels and to retain all information we concatenate them. This is in contrast to the Deep Jet architecture [8], where feature channels from all branches have the same semantic meaning, hence it is meaningful to add them.

The MS FCN architecture addresses the influence of features of different scales on the reward as well as the possibility for spatial invariance with high level features while retaining the precise location information offered by the low level features. Furthermore, all architectures are designed such that the size of the receptive field for each location in the final cost-map is ensured to be large enough to encapsulate the area occupied by the vehicle plus additional surrounding terrain.

In the next section, we evaluate all three architectures qualitatively and quantitatively, and compare them with a carefully manually designed cost-function that is currently deployed on our research platforms.

IV. Experiments

A. Large-Scale Data Collection

We demonstrate the applicability of the proposed approach to complex, real-world, urban driving environments on a large dataset that was collected over the course of one year involving 13 different drivers driving manually every two weeks on the pedestrian walkways and cycle lanes in the city of Milton Keynes, UK.

Our data collection platform is a modified GEM (Global Electric Motorcars) golf cart (cf. Figure 3), equipped with a

Fig. 3: Our research mobile platform – a modified GEM golf cart.

variety of sensors including 2D as well as 3D LIDARs and stereo- and mono-cameras. The relevant sensors for this work include two Velodyne HDL-32E scanners and a Bumblebee XB3 stereo camera.

The dataset collected contains a variety of challenging obstacles such as trees, bollards, bike racks, curbs, stairs and pedestrians. An additional challenge of the dataset lies in the variation in driving behaviours and styles due to the multiple drivers. We extracted in total over 25,000 trajectories, each about 15m long, from the more than 120km of driving contained in the dataset. The extracted trajectories are then randomly divided into a training set containing 95% of the data and a test set based on the remaining trajectories.

B. Input Representations

The input data to our network is based on pointclouds measured by the two 3D Velodyne scanners with overlapping field-of-view. Given the calibration of the LIDARs to the robot frame, the resulting 3D pointclouds are mapped into a 2D-grid based static map on the ground plane to enable the application of FCNs to the environment representation. The extracted statistics include mean height, height variance and a binary indicator if the cell is visible in any scan. The grid has a size of 25m x 25m and a resolution of 0.25m per cell.

C. Demonstration Trajectories

Figure 4 shows an example of gathered trajectories based on demonstration driving data. As a first step a serial chain of transforms is extracted from the motion of the robot estimated from Visual Odometry [13] on stereo images from the Bumblebee XB3 camera. The extracted transform chain is subsequently mapped into the static map frame and discretised to fit into the grid representation. Repeated states are removed from the sequences before conversion into state-action trajectories through the MDP to prevent overemphasis

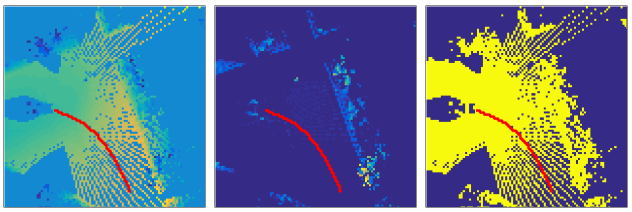


Fig. 4: Visualisation of demonstration trajectories (in red) on all fields of a static map. From left to right: mean height, height variance, cell visibility.

of keep-position actions. Finally, the action space is simplified to a discrete set of motions around the current position. The simplification is necessary to keep the problem tractable and use the given environment representation based on a 2D grid necessary for convolutions.

D. Implementation Details

Following each non-linearity we insert a layer of Batch Normalisation (BN) [14] to reduce internal co-variate shift within batches and speed up training, while adding to model regularisation - additionally addressed with elastic net loss on the network parameters - and increasing generalisation performance.

To increase the number of training samples, the set includes partially overlapping trajectories with static maps from different positions. While the trajectory samples can be seen as not fulfilling the i.i.d. criterion, the advantage of additional input representations from different view points – the static maps – leads to significant performance gain after randomisation of the data sequences. A principal reason for this fact is that feature representations based on LIDAR scans differ significantly according to the distance away from the sensor as laser points become sparser as the distance from the sensor increases. Hence different viewpoints of the same environment lead to varying feature representations.

E. Evaluation

To evaluate the accuracy of approximating human driving behaviour, we use two metrics: the negative log-likelihood of the demonstration data (NLL) as well as the Modified Hausdorff Distance (MHD) [16]. Both metrics evaluate model performance for predicting demonstrator behaviour on the test set with the latter being used to determine a metric of distance between the original demonstration trajectories and new samples generated based on the cost-function and resulting policy of the trained system. The presented metrics result from averaging over the metrics for 1000 test trajectories and in case of the MHD 10 samples per single test.

As part of the following sections, we evaluate the different models against the handcrafted cost-function currently applied in the planning module. The main idea behind the design of this cost-function, as illustrated in Figure 5, is that obstacles generally represent areas of large height variance in each cell and areas without LIDAR information are labelled as unknown, which in the classification task is treated as not traversable. Furthermore, obstacles are extended by the

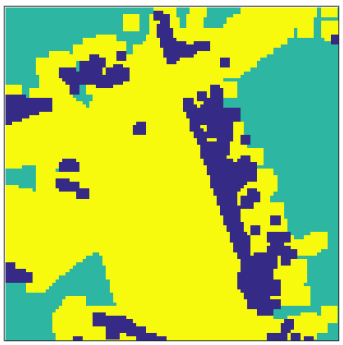


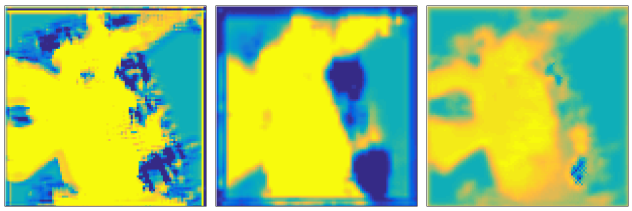
Fig. 5: Handcrafted cost-function (our baseline for comparison) for the same scenario shown in Figure 4.

radius of the smallest circle encapsulating the GEM platform (Minkowski sum).

Prediction metrics	Standard FCN	Pooling FCN	MS FCN	Manual CF
NLL	69.35	69.73	65.39	78.13
MHD	0.221	0.230	0.200	0.284

TABLE I: Prediction performance on the test set given by Negative Log-Likelihood (NLL) and Modified Hausdorff Distance (MHD).

While the standard FCN already predicts test trajectories significantly better than the manual cost-function (see Table I) and therefore describes human driving behaviour more accurately, the qualitative analysis of the demonstration maps in Figure 6 shows that for areas where LIDAR scans get sparser the network has problems reasoning about the true reward. The use of the pooling architecture without parallel information branches performs better in terms of a smooth cost-function when interpreting the same areas. Due to the Max-Pooling layer the approach is able to infer traversable terrain. However, it also leads to a loss of information in the pooling step as discussed in section III-C. Best performance is achieved when spatial invariance in one branch is combined with the preservation features in a parallel chain of the DAG-like MS-FCN architecture.



(a) (b) (c)

Fig. 6: Visualisation of the learned cost-maps based on (a) Standard FCN, (b) Pooling FCN, and (c) MS FCN.

We furthermore perform a quantitative analysis of the resulting cost model by evaluating the binary classification of trajectories as collision/free space with the results presented in Table II. Since we try to learn the unspecified cost-function

that underlies the mutual behaviour of multiple drivers, there is no ground-truth in costs. However, the driven trajectories themselves present knowledge about the actual environment and every driven trajectory is clearly free of collisions.

The classification of trajectories based on learned cost-functions needs a threshold to decide for the minimal cost at which a collision occurs and the trajectory can be classified as untraversable. To enable the determination of such a threshold we introduce artificial trajectories representing collisions into the evaluation process such that we can determine Type I (false positives) and Type II errors (false negatives) for the classification. The artificial trajectories were chosen based on direct information from LIDAR as well as camera data that ensures collision. The final decision for a threshold was made to reduce FPR to 0%. It can be preferable to increase the threshold slightly such that the FNR drops significantly and the cost-function enables planning to find more traversable paths. This leads however to an increase in FPR and should be constrained by setting a maximum limit.

Error metrics	Standard FCN	Pooling FCN	MS FCN	Manual CF
FNR	0.471	1.000	0.206	0.441
FPR	0.000	0.000	0.000	0.000

TABLE II: Trajectory evaluation performance.

With the manual cost-function design being strictly conservative with respect to the traversability of terrain it results in 0 % false positives rate (FPR) while showing a significant number of false negatives. The adaption of cost-function thresholds for 0% FPR turns out to be complicated. While it is often easy to achieve rates of under 5% the final adjustment significantly increases the false negatives rate. In case of the pooling architecture this rendered it infeasible to continue planning with 0% FPR, since a FNR of 100% represents that none of the paths are classified as traversable. The MS FCN which combines the benefits of both other models (see section III-C) performs best with a reduction in FNR of about 50% in comparison to the manual cost-map.

F. Robustness Study on Systematic Biases

Learning a cost-function and its most relevant features proves robust towards limited systematic flaws in the configuration of the robot. An inaccurate calibration for example can lead to complete failure for a manually crafted cost-function as presented in Figure 7. The artificial obstacles in front of the vehicle can be created due to an imprecise calibration of the pitch angle between the platform and one of the LIDARs of as little as 1°. Due to this introduced perturbation in pitch angle of the right Velodyne, the manually defined cost-map creates obstacles in this instance as the height variance of points in a specific cell increases significantly with rising distance from the vehicle.

However, the learned cost-map is able to handle this problem and learn that driving is possible over this terrain. As long as the input representation is rich enough to distinguish between the new features describing artificial obstacles and real walls etc. the system is able to differentiate between features representing obstacles that we have never traversed in any demonstration trajectories and acceptable terrain which was often shown to be traversable in the training data.

Taking the metrics introduced in Section IV-E, we evaluate the performance of the MS FCN architecture versus the manually defined cost-function, clearly showing the benefits in using a trained system in Tables III and IV.

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, though compares worse than the case with the correct calibration (cf. Table II), still remains within functional range.

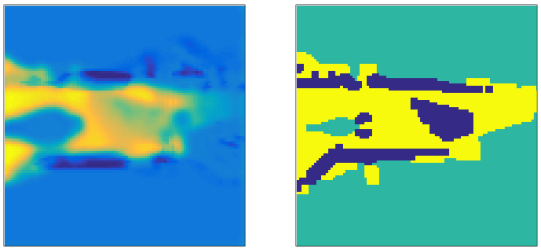


Fig. 7: Example cost-maps based on miscalibrated data with MS FCN (left), and the handcrafted cost-function (right).

Error metrics	MS FCN	Manual CF
FNR	0.525	0.971
FPR	0.000	0.000

TABLE III: Trajectory evaluation performance on miscalibrated data.

Prediction metrics	MS FCN	Manual CF
NLL	69.35	89.40
MHD	0.267	0.432

TABLE IV: Prediction performance on miscalibrated data.

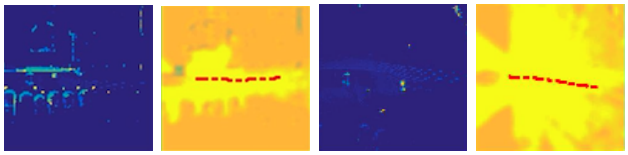
G. Discussion

We argue that given the large amount of training data and complex features, only a high-capacity, parametric approach to IRL possesses the capability to approximate the reward mapping while being computationally tractable. Handcrafted features of the required complexity can only be designed with significant expert knowledge of the domain, perception pipeline and expected behaviour. As seen in Section IV, features describing domain specific obstacles can be easily missed in the preprocessing setup and are inherently hard to design. Spatial features as learned by an FCN on the other hand are inherently optimised for the task. Robustness with respect to unknown but systematic perception inaccuracies as in Section IV-F is generally beyond reach since the feature construction would depend on knowledge of the type of feature resulting from e.g. miscalibration. Furthermore, while nonparametric approaches such as GPIRL might possess the capacity to approximate complex nonlinear reward functions when given these hypothetically perfect features, the

amount of training data severely influences the feasibility of such an approach - potentially rendering it completely intractable. While showing great performance on prediction and classification tasks, one consistent aspect of all learned cost-functions represented in Figure 6 is the smoothing of the cost-function around obstacles, which is caused since training focuses on features describing traversable terrain. Since most driving in the demonstrations happens around the middle of paths, the features are optimised to represent these areas. During test time, the agent can encounter a significantly different sensor data distribution. While IRL generalises preferably to direct imitation, this problem can be additionally addressed by transferring a DAgger-like [17] approach from behavioural cloning to IRL.

The current approach focuses on generating a cost map with respect to static environment aspects as represented in Figure 8. This is deemed advantageous as the planning module separates into determining spatially feasible paths and subsequent speed-profiling with respect to dynamic obstacles to reduce the dimensionality of the problem.

While this spatio-temporal separation fulfils the requirements of the current motion planning module, future research will address the additional learning of the driver's preference for specific speed profiles in presence of dynamic obstacles. This will require the processing of multiple sensor inputs from different times instead of focusing on from a limited period around the start of each trajectory, since the position of dynamic obstacles will change during the traversal of a given trajectory. A natural extension here would be stacking of these sensor inputs or sequential processing via a recurrent module.



(a) Variance (b) Cost-map (c) Variance (d) Cost-map

Fig. 8: Example for removal of pedestrians – represented by bright spots in the height variance map (a&c) – from static cost maps (b&d).

V. Conclusion and Future Work

In this work we apply the framework of Maximum Entropy Deep Inverse Reinforcement Learning [4] to large scale urban navigation scenarios with over 25k demonstration samples collected from 13 different drivers. Based on the extensive dataset of over 120 km of driving, the high capacity of FCNs enables us to deduce the operator's underlying reward mapping directly from sensory input. This is possible even under systematic perturbations of the robot configuration which render handcrafted cost-functions completely impracticable.

We develop a multi-scale network architecture capable of capturing relevant features at different scales and outperform the manual cost-function in prediction performance as well as the classification accuracy for test trajectories. Due to working directly on the sensory input and learning spatial features, the approach is able to correct for systematic biases which we demonstrate by application to a miscalibrated dataset.

Future work will target the investigation of learning from negative demonstration samples (collisions), off-road driving behaviour and extension towards dynamic obstacles as discussed in Section IV-G.

References

[1] Chris Urmson, Joshua Anhalt, Drew Bagnell, Christopher Baker, Robert Bittner, MN Clark, John Dolan, Dave Duggins, Tugrul Galatali, Chris Geyer, et al. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466, 2008.

[2] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, et al. Stanley: The robot that won the DARPA Grand Challenge. *Journal of Field Robotics*, 23(9):661–692, 2006.

[3] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1. ACM, 2004.

[4] Markus Wulfmeier, Peter Ondruska, and Ingmar Posner. Deep inverse reinforcement learning. *CoRR*, abs/1507.04888, 2015.

[5] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, pages 1433–1438, 2008.

[6] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[7] Henrik Kretzschmar, Markus Spies, Christoph Sprunk, and Wolfram Burgard. Socially compliant mobile robot navigation via inverse reinforcement learning. *The International Journal of Robotics Research*, page 0278364915619772, 2016.

[8] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[9] Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. *CoRR*, abs/1603.00448, 2016.

[10] 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.

[11] Yoshua Bengio, Yann LeCun, et al. Scaling learning algorithms towards AI. *Large-scale kernel machines*, 34(5), 2007.

[12] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[13] Winston Churchill and Paul Newman. 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)*, pages 1371–1376. IEEE, 2012.

[14] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[15] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.

[16] Kris M. Kitani, Brian D. Ziebart, James Andrew Bagnell, and Martial Hebert. Activity Forecasting. In Andrew Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid, editors, *Computer Vision ECCV 2012*, number 7575 in Lecture Notes in Computer Science, pages 201–214. Springer Berlin Heidelberg, 2012.

[17] Stéphane Ross, Geoffrey J Gordon, and J Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the International Conference on Artifical Intelligence and Statistics*, 2010.