Learning from Demonstration in the Wild

Feryal Behbahani[1], Kyriacos Shiarlis[1], Xi Chen[1], Vitaly Kurin[1,2], Sudhanshu Kasewa[1,2], Ciprian Stirbu[1,2],
João Gomes[1], Supratik Paul[1,2], Frans A. Oliehoek[1,3], João Messias[1], Shimon Whiteson[1,2]

***Abstract*— *Learning from demonstration* (LfD) is useful in settings where hand-coding behaviour or a reward function is impractical. It has succeeded in a wide range of problems but typically relies on manually generated demonstrations or specially deployed sensors and has not generally been able to leverage the copious demonstrations available *in the wild*: those that capture behaviours that were occurring anyway using sensors that were already deployed for another purpose, e.g., traffic camera footage capturing demonstrations of natural behaviour of vehicles, cyclists, and pedestrians. We propose *video to behaviour* (ViBe), a new approach to learn models of behaviour from unlabelled raw video data of a traffic scene collected from a single, monocular, initially uncalibrated camera with ordinary resolution. Our approach calibrates the camera, detects relevant objects, tracks them through time, and uses the resulting trajectories to perform LfD, yielding models of naturalistic behaviour. We apply ViBe to raw videos of a traffic intersection and show that it can learn purely from videos, without additional expert knowledge.**

I. Introduction

Learning from demonstration (LfD) is a machine learning technique that can learn complex behaviours from a dataset of expert trajectories, called *demonstrations*. LfD is particularly useful in settings where hand-coding behaviour, or engineering a suitable reward function, is too difficult or labour intensive. While LfD has succeeded in a wide range of problems [1], [2], [3], nearly all methods rely on either artificially generated demonstrations (e.g., in laboratory settings) or those collected by specially deployed sensors (e.g., MOCAP). These restrictions greatly limit the practical applicability of LfD, which to date has largely not been able to leverage the copious demonstrations available *in the wild*: those that capture behaviour that was occurring anyway using sensors that were already deployed for other purposes.

For example, consider the problem of training autonomous vehicles to navigate in the presence of human road users. Since physical road tests are expensive and dangerous, simulation is an essential part of the training process. However, such training requires a realistic simulator which, in turn, requires realistic models of other agents, e.g., vehicles, cyclists, and pedestrians, that the autonomous vehicle interacts with. Hand-coded models of road users are labour intensive to create, do not generalise to new settings, and do not capture the diversity of behaviours produced by humans.

LfD is an attractive alternative. In principle, subjects could be recruited to demonstrate such behaviour or existing road users could be augmented with sensors to record their trajectories. However, doing so would be expensive and yield only limited datasets. A more effective way would be to use the abundance of relevant demonstrations available in the wild, such as traffic camera footage. Unfortunately, there are currently no LfD methods that can learn from such sources of traffic demonstrations.

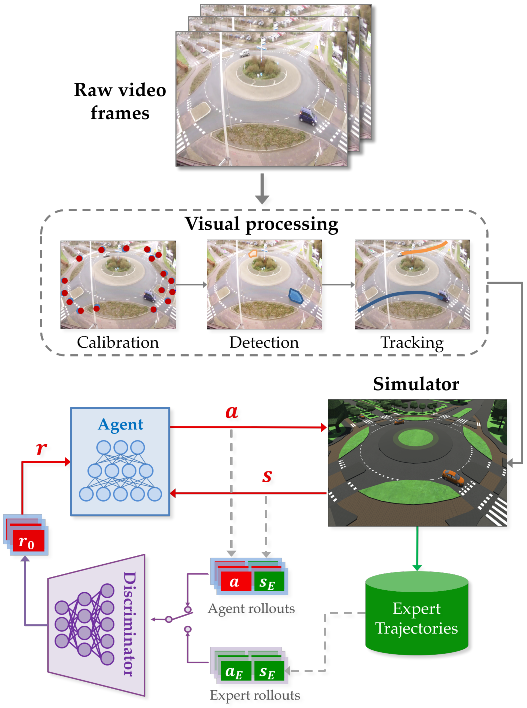

Fig. 1. Schematic of the ViBe approach

In this paper, we propose *video to behaviour* (ViBe), a new approach to learn models of road user behaviour from unlabelled raw video data of a traffic scene collected from a single, monocular, initially uncalibrated camera with ordinary resolution. Our approach, illustrated in Figure 1, works by calibrating the camera (using available satellite images), detecting the relevant objects, and tracking them through time. Each trajectory, together with the static and dynamic context of that road user at each moment in time, is then fed as a demonstration to our LfD system, which can learn robust behaviour models for road users. The resulting models are then used to populate a simulation of the scene built using the Unity game engine.

The contributions of this paper are two-fold: First, we present a vision pipeline that can track different road users

[1] Latent Logic, Oxford, England
[2] University of Oxford, Oxford, England
[3] Delft University of Technology, Delft, Netherlands

and map their tracked trajectories to 3D space and is competitive with the state-of-the art approaches for image space tracking. Second, we extend *generative adversarial imitation learning* (GAIL) [4], a state-of-the-art LfD method, with a novel curriculum-based training regime that enables our agents to gradually learn to mimic temporally extended expert demonstrations and successfully generalise to unseen situations. We evaluate our method against several baselines, including *behavioural cloning* (BC) and state-of-the-art variants of GAIL. Using a number of metrics, we show that our method can better imitate the observed demonstrations and results in more stable learning.

II. Related Work

A. Computer Vision

In recent years, neural network approaches have significantly advanced the state of the art in computer vision tasks such as classification [5] and object detection [6]. Object detection is usually performed using region-based object detectors such as Fast R-CNN [7], Faster R-CNN [8], or Mask R-CNN [9]. Such methods are usually slower but more accurate than single-object detectors such as SSD [10], YOLO [11], RetinaNet [12], and hence more appropriate for the application considered here.

When tracking multiple objects, *tracking by detection*, in which objects are first detected, then associated into tracks, is usually preferred. State-of-the art tracking methods employ deep features [13], [14] often generated by Siamese networks [15], [16] alongside image space motion models [17] and *intersection over union* (IOU) trackers [18].

Our work employs a number of techniques for robust detection and tracking. However, unlike most vision pipelines, ours maps detections to 3D space, and makes extensive use of 3D information while tracking. Recent work [19] explores a similar application and uses the resulting 3D trajectories to estimate car velocities and detect traffic anomalies. By contrast, we use the trajectories as input to LfD.

B. Learning from Demonstration

ViBe's LfD component extends GAIL [4] which is inspired by inverse reinforcement learning [20], [21], [22] and is discussed further in Section III. A wide range of LfD techniques have been developed using supervised, unsupervised, or reinforcement learning [2]. However, most methods [3], [23], [24], even when using raw video as sensory input [25], rely on either artificially generated demonstrations or those collected by specially deployed sensors, limiting their application in realistic domains.

By contrast, ViBe leverages demonstrations of behaviour that was occurring naturally. The same idea has been used to imitate basketball teams [26], predict taxi driver behaviour [27], and control complex animations [28]. However, all these methods still rely on sensors (or manual labelling) that provide ground truth information about the observed demonstrations, whereas ViBe extracts trajectories directly from raw, unlabelled videos; the satellite images used for calibration are the only external input required.

Related to ViBe are several existing LfD methods that learn road and pedestrian behaviour [29], [30], [31], [32]. Most relevant is learning highway merging behaviour [33], [34] from NGSIM [35], a publicly available dataset of vehicle trajectories. However, these methods again rely on manual labelling, synthetic data or specialised equipment to obtain the trajectories, while ViBe learns from raw, unlabelled videos of behaviour.

Recent work proposed a method that can learn to play ATARI games by observing YouTube videos [36]. Like ViBe, this method leverages raw videos, and existing publicly available data. However, it trains only a single agent operating in 2D space, whereas ViBe learns to control multiple interacting agents in 3D space.

Concurrently to our work, Peng et al. [37] proposed a similar approach in the context of character animation. An off-the-shelf vision module extracts 3D poses from unstructured YouTube videos of single agents performing acrobatic motions. A simple LfD approach then rewards behaviour that matches waypoints in individual demonstrations. By contrast, we consider a more challenging setting with multiple agents, occlusions, and complex interactions between agents. Consequently, behaviour detection, reconstruction, and imitation are more difficult. In particular, interactions between agents preclude a waypoint-matching approach, as there is no unique set of waypoints for an agent to match that would be robust to changes in other agents' behaviour.

III. Background

To realistically model the traffic environment of an autonomous vehicle, we need to simulate multiple agents interacting in the same environment. Unfortunately, due to the large number of road users that may populate a traffic scenario, learning a centralised policy to control all agents simultaneously is impractical. The size of the joint action space of such a policy grows exponentially in the number of agents, leading to poor scalability in learning. Furthermore, it is crucial to model variable numbers of agents (e.g., cars routinely enter and leave an intersection), to which such *joint policies* are poorly suited (each agent typically has a fixed agent index).

To this end, we take an approach similar to that of *independent Q-learning* (IQL) [38], where each agent learns its own policy, conditioned only on its own observations. The other actors are effectively treated as part of the environment. We can then treat the problem as one of single-agent learning and share the parameters of the policy across multiple agents. Parameter sharing [39] avoids the exponential growth of the joint action space and elegantly handles variable numbers of agents. It also avoids instabilities associated with decentralised learning by essentially performing centralised learning with only one policy.

We model the problem as a *Markov decision process* (MDP). The MDP is defined by the tuple $(\mathcal{S}, \mathcal{A}, P, R)$. $\mathcal{S}$ represents the set of environment states, $\mathcal{A}$ the set of actions, $P(s_{t+1}|s_t, a_t)$ the transition function, and $R(s_t, a_t)$ the reward function. We use π for the stochastic policy learnt by our agent and π_E for the expert policy which we can access only through a dataset $\mathcal{D}_E$. The agent does not have access to $R(s_t, a_t)$ and instead must mimic the expert's demonstrated behaviour. Given a dataset $\mathcal{D}_E$, we denote sample trajectories as τ^E. They consist of sequences of observation-action pairs generated by the expert $\tau^E = \{(s^E_1, a^E_1), \ldots, (s^E_T, a^E_T)\}$. Similarly, we denote trajectories generated by our agent as $\tau = \{(s_1, a_1), \ldots, (s_T, a_T)\}$. In our case, $\mathcal{D}_E$ is obtained from raw videos, via the process described in Section IV.

The simplest form of LfD is *behavioural cloning* (BC) [40], [41], which trains a regressor (i.e., a policy) to replicate the expert's behaviour given an expert state. BC works well for states covered by the training distribution but generalises poorly due to compounding errors in the actions, a problem also referred to as *covariate shift* [42]. By contrast, GAIL [4] avoids this by learning via interaction with the environment, similar to *inverse reinforcement learning* [20] methods.

GAIL aims to learn a deep neural network policy π_θ that cannot be distinguished from the expert policy π_E. To this end, it trains a *discriminator* D_ϕ, also a deep neural network, to distinguish between state-action pairs coming from expert and agent (a process similar to GANs [43]). GAIL optimises π_θ to make it difficult for the discriminator to make this distinction. Formally, the GAIL objective is:

$$\min_\theta \max_\phi \mathop{\mathbb{E}}_{\pi_\theta} \left[\log(D_\phi(s,a))\right] + \mathop{\mathbb{E}}_{\pi_E} \left[\log(1 - D_\phi(s^E, a^E))\right] .$$

Here, D_ϕ outputs the probability that (s, a) originated from π_θ. As the agent interacts with the environment using π_θ, (s, a) pairs are collected and used to train D_ϕ. Then, GAIL alternates between a gradient step on ϕ to increase the objective function with respect to D, and an RL step on θ to decrease it with respect to π. Optimisation of π can be done with any RL algorithm using a reward function of the form $R(s, a) = -\log(D_\phi(s, a))$. Typically, GAIL uses policy gradient methods that approximate the gradient with Monte Carlo rollouts [44] or a critic [45]. Optimisation of D_ϕ minimises a cross entropy loss function.

Early in training, the state-action pairs visited by the policy are quite different from those in the demonstrations, which can yield unreliable and sparse rewards from D_ϕ, making it difficult to learn π_E. We show how we address this problem by introducing a novel curriculum in Section IV-C.

In multi-agent situations, GAIL agents trained in a single-agent setting can fail to generalise to multi-agent settings [34]. PS-GAIL [34] is an extension to GAIL that addresses this issue by gradually increasing the number of agents controlled by the policy during training. We compare to PS-GAIL experimentally in Section V. However, it is complementary to the Horizon GAIL method we propose in Section IV-C and future work can focus on using them in conjunction.

IV. ViBe: Video to Behaviour

In this section, we describe ViBe, which learns road behaviour policies from raw traffic camera videos (see Figure 1). We first describe how trajectories are extracted from these videos. We then describe how they are used to create a simulation of the scene. Finally, we detail how the trajectories and the simulator are used to learn realistic road behaviour policies via our novel LfD approach.

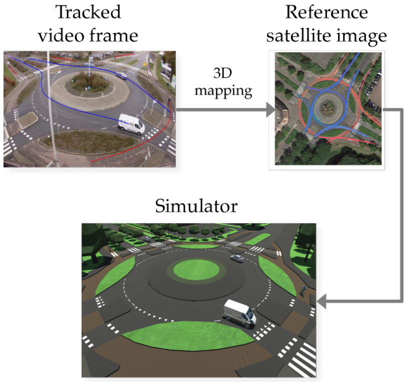

Fig. 2. Example of how ViBe's vision module tracks cars (blue) and pedestrians (red). The tracks are projected to 3D space using a reference satellite image from Google Maps. The tracks are played back in a simulation of the scene developed in Unity.

A. Extracting Demonstrations

This section describes our vision pipeline, whose main steps are *detection*, *calibration*, and *tracking*. Our detector uses the bounding box output of a pre-trained model of Mask R-CNN [9] [6] based on the ResNet-101 [5] architecture, pre-trained on the COCO dataset [46]. Since we are only interested in the traffic information, we remove all classes except car, bus, truck, pedestrian, and bicycle.

The next step is calibration. As traffic cameras tend to have a large field of view, the camera images tend to be highly distorted. As we do not have access to the cameras, we are unable to calibrate the camera using traditional methods (e.g., using a checkboard) [47]. Instead, we obtain a top-down satellite image of the scene from Google Maps and add landmark points to both camera and satellite images. We then undistort the camera image and use the landmark points to calculate the camera matrix. Given the camera calibration we map the detected bounding boxes into 3D by assuming that the detected object is a fixed height above the ground, with the height depending on its class.

The final step is tracking multiple objects in unstructured environments. Our multiple object tracking (MOT) module is similar to that of Deep SORT [14], which makes use of an *appearance model* to make associations. For each scene, we train an appearance model using a *Siamese network* (SN) [15]. We first run our object detector over the whole video, followed by an IOU tracker. This yields short tracks that we call *tracklets*. Objects in the same tracklets form positive pairs, and objects from different tracklets form negative pairs used to train the SN. To avoid the possibility of similar objects appearing in negative pairs, we form these pairs using tracklets with a large temporal difference. The SN is trained using a cosine distance metric and a contrastive loss.

Our MOT pipeline then processes the detected objects through several steps. Track initialisation occurs when a simple IOU tracker associates more than five consecutive detections. The initialised track is mapped to 3D space, where a Kalman filter predicts the next position of the object. Next, objects in the next frame within the vicinity of this prediction are compared with the current track using the features generated by the SN. An association is made if this comparison yields a cosine distance in the feature space below a certain threshold. If no such association is made, the tracker attempts to associate detections using IOU. If association still fails, a final attempt is made using nearest neighbour association in 3D. Figure 2 shows an example output of our tracking pipeline in both 2D and 3D space.

B. *Simulation*

Our vision pipeline outputs timestamped trajectories of different road users. However, a simulator also requires a reliable representation of the static elements of the scene such as pavements and zebra crossings. To this end, we use Google Maps as a reference to build a simulation of the scene in Unity. Building the static elements of the simulation is straightforward and significantly easier than realistically modeling the dynamic elements of the scene. In this paper, we simulate a roundabout intersection in Purmerend, a city in the Netherlands that provided the traffic video data used in our experiments. Figure 2 shows how the scene with some tracks from our vision pipeline is recreated in our simulator.

Section IV-C describes our LfD approach, which requires a state representation for the agent. Our simulator generates such a representation based on both the static and dynamic context. Pseudo-LiDAR readings, similar to those in [34], are used to represent different aspects of the static (e.g., zebra crossings and roads) and dynamic (e.g., distance and velocity of other agents) context of the agent. In addition, we provide information such as the agent's heading, distance from goal, and velocity. Our simulator uses a simple linear motion model, which we found sufficient for learning, though in the future individual motion models for each road entity could be considered.

Given a start frame in the dataset, our simulator plays back tracked trajectories from that frame onwards, produces observations, and accepts actions from agents controlled by neural network policies. In other words, it provides exactly the environment needed to both perform LfD on the extracted trajectories and evaluate the resulting learnt policies.

C. *Learning*

Given the trajectories extracted by the vision processing from Section IV-A, ViBe uses the simulator from Section IV-B to learn a policy that matches those trajectories. Learning is based on GAIL, which leverages the simulator to train the agent's behaviour for states beyond those in the demonstrations, avoiding the compounding errors of BC. However, in the original GAIL method, this interaction with the simulator means that the agent has control over the visited states from the beginning of learning. Consequently, it is likely to take bad actions that lead it to undesirable states, far from those visited by the expert, which in turn yields sparse rewards from the discriminator and slow agent learning.

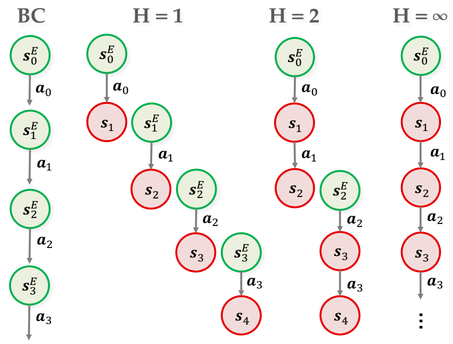

Fig. 3. Schematic of Horizon GAIL for different values of the horizon H, compared to BC. Green circles indicate bootstrapped expert states, red circles correspond to states that the agent encounters after acting in the environment. When $H = \infty$, Horizon GAIL matches original GAIL.

To address this problem, we propose *Horizon GAIL*, which, like BC, bootstraps learning from the expert's states, in this case to ensure a reliable reward signal from the discriminator. To prevent compounding errors, we use a novel horizon curriculum that slowly increases the number of timesteps for which the agent interacts with the simulator. Thus, only at the end of the curriculum does the agent have the full control over visited states that the original GAIL agent has from the beginning. This curriculum also encourages the discriminator to learn better representations early on.

In each episode, the agent is initialised from a random expert state, s_t^E and must act for H steps, where H is the length of the horizon. Once the horizon is reached, the simulation ends but the episode is not considered terminated. Instead, Horizon GAIL uses an actor-critic approach, with the actor following a gradient estimated from an n-step return, with $n = H$, bootstrapping from a critic V_ψ when the horizon is reached. This prevents the agent from learning myopic behaviour when H is small. Hence, while GAIL is agnostic about the policy gradient method it uses, Horizon GAIL requires a critic in order to bootstrap beyond the simulated horizon.

When $H = 1$, Horizon GAIL is similar to BC. In fact, pre-training GAIL with BC is known to be beneficial [33], [30], [48], [49]. However, even with $H = 1$, a crucial difference remains (see Figure 3). BC does not interact with a simulator, as the agent simply learns to predict the expert's action given its state. By contrast, when $H = 1$, the Horizon GAIL agent's action is fed back into the simulator, which generates s_{t+1} and the policy gradient estimate bootstraps with $V_\psi(s_{t+1})$. When $H = 2$, the agent, initialised from s_t^E, acts for two steps in the simulator before being terminated. H is increased during training according to a schedule. When $H = \infty$, Horizon GAIL is equivalent to GAIL.

Gradually moving from single step state-action pairs to more difficult multi-step trajectories helps stabilise learning. It allows the generator and discriminator to jointly learn to generalise to longer sequences of behaviour and match the expert data more closely while ensuring the discriminator does not collapse early in training. We found that Horizon GAIL was critical to successfully reproduce naturalistic behaviour in our complex traffic intersection problem, as we show in Section V-B.

V. Experimental results

We evaluate ViBe on a complex multi-agent traffic scene involving a roundabout in Purmerend (Section IV-B). The input data consists of 850 minutes of video at 15 Hz from the traffic camera observing the roundabout. Our vision pipeline identifies all the agents in the scene (e.g., cars, pedestrians and cyclists), and tracks their trajectories through time, resulting in around 10000 car trajectories. Before any learning, these trajectories are filtered and pruned. Specifically, any trajectories that result in collisions or very large velocities are considered artifacts of the tracking process and are not used during training. We split the resulting dataset into training, validation, and test sets such that there is no temporal overlap, i.e., no test trajectories occur at the same time as training trajectories. The validation set is used to tune hyperparameters and choose the best performing model (for all baselines) in evaluation. As discussed in Section IV-B, we can use our simulator to play back these trajectories at any point in time (see Figure 1).

When training with Horizon GAIL, in each episode the agent is initialised at a point sampled from an expert trajectory. The sampled point determines the full initial state of the simulator, including position, velocity, and heading of all agents in the scene. We use our policy to simulate the agent for H steps. The agent is also assigned a goal corresponding to the last state of the expert trajectory. The episode terminates if the agent collides with an object or another agent, or reaches its goal.

We compare Horizon GAIL to a number of baselines: BC, GAIL [4] and PS-GAIL [34], using the same dataset and observation and action spaces to train all methods. We show results using the best hyperparameters we found after tuning them separately for each method.

Policies, π_θ, take as input 64 dimensional pseudo-LiDAR observations with a field of view of 2π radians, generated by our simulator as described in Section IV-B. These LiDAR observations are stacked together and processed in two layers of 1x1 convolutions of 15 and 3 channels respectively. These convolutions act as channel mixing operations but maintain the spatial information of the original signal. The output then passes through a series of fully connected layers and is concatenated with the agent's orientation, distance from the goal, and a one-hot encoding of the target roundabout exit. The network outputs displacements in Cartesian coordinates, used by the simulator to update the agent's location.

We use identical core architectures for the discriminator D_ϕ and value function V_ψ. Unlike [34], we do not represent the policy using a recurrent neural network, as we found that a feedforward network worked well in practice.

TABLE I

Comparison of ViBe vision module to baseline trackers

	NT	IDF1	IDP	IDR
IOU	400	51.1%	50.3%	51.8%
Deep SORT	129	68.1%	66.6%	69.7%
ViBe	**97**	**70.5%**	**68.1%**	**73.1%**

We train π_θ with *proximal policy optimisation* (PPO) with a clipping objective [45], an actor-critic method known to perform well for long-horizon problems [50]. We train each model for 5000 epochs, each containing 1024 environment interactions. For Horizon GAIL, the horizon schedule starts with $H = 1$ and increments by 1 every 100 epochs. However, performance is quite robust to this hyperparameter: varying the schedule from 50 to 200 epochs did not create any significant performance differences.

A. Performance Metrics

To evaluate the ViBe vision module, we measure the reliability of the tracks it generates using the metrics introduced by Ristani et al. [51]: number of tracked trajectories (NT), identity F1 score (IDF1), identity precision (IDP) and identity recall (IDR). These metrics are suitable because they reflect the key qualities of reliably tracked trajectories.

To evaluate our policies, we chose a 4000 timestep window of the test data and simulated all the cars within that interval. These windows do not overlap for each evaluation run. Pedestrians and other road users are played back from the dataset. In contrast to training, during evaluation we do not terminate the agents upon collision, so as to assess how well each method models long term behaviour.

Unlike in reinforcement learning, where the true reward function is known, performance evaluation in LfD is not straightforward and typically no single metric suffices. Several researchers have proposed metrics for LfD, which are often task specific [33], [52], [48]. We take a similar approach, using a suite of metrics, each comparing a different aspect of the generated behaviour to that of human behaviour.

During evaluation we record the positions and velocities of all simulated agents. Using kernel density estimation, we estimate probability distributions for speed and 2D space occupancy (i.e., locations in 2D space) as well as a joint distribution of velocities and space occupancy. The same distributions are computed for the ground truth data. We then measure the Jensen-Shannon divergence (JSD) between the data and the respective model generated distributions for these three quantities. We also measure how often the simulated agents collide with objects or other agents in the environment, i.e., the collision rate. Finally, we measure how often the agents fail to reach their goal.

B. Results

To validate the ViBe vision module, we manually label 43 trajectories from the dataset and then compare its performance against two baselines, a simple IOU [18] tracker and Deep SORT [13], [14], a state-of-the-art MOT pipeline. We replace Deep SORT's appearance model with our own, as it is specifically trained for this scene.

The results in Table I show that the ViBe vision module outperforms both baselines. In particular, ViBe's higher IDF1 score gives confidence that the trajectories provided are of sufficient quality for LfD. The most substantial difference between Deep SORT and ViBe is that ViBe performs Kalman

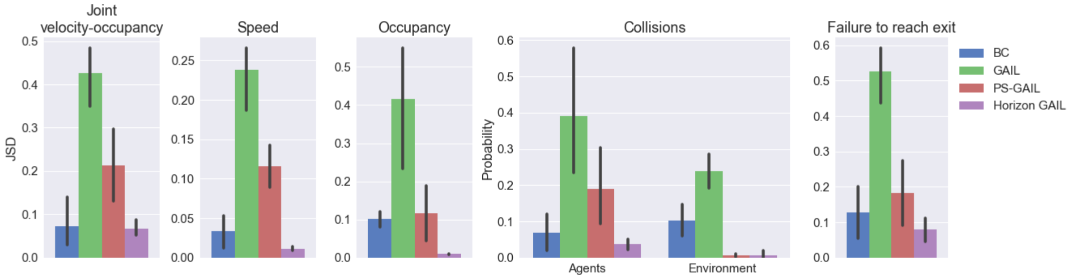

Fig. 4. Results of evaluation across 4 independent 4000 timesteps of multi-agent simulations across different metrics: Jensen-Shannon divergence between joint velocity-occupancy, speed and occupancy distributions of ground truth and simulated agents. The collision probability, either with other agents or the environment. Probability of failing to reach the correct exit.

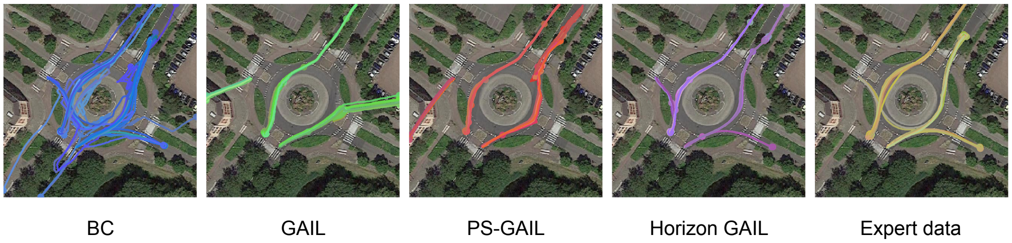

Fig. 5. Top views of the trajectories taken by the agents, when trying to replicate the expert trajectories shown on the right-most column. These trajectories are produced across 4000 timesteps of multi-agent simulation.

filtering in 3D space, which likely explains the performance difference. Even for ViBe, the number of tracked trajectories (NT) is substantially higher than ground truth (43). However, this is not caused by false trajectories but merely by the tracker splitting a single trajectory into separate ones. This in turn implies that ViBe produces longer tracks than the baseline methods.

The results of our LfD evaluation can be seen in the following figures: Figure 4 shows performance with respect to the evaluation metrics discussed in Section V-A for 4 independent 4000 timesteps of multi-agent simulations. Figure 5 shows the trajectories generated by a single such simulation by each method. Horizon GAIL outperforms all baselines and produces trajectories that more closely resemble the data. GAIL and PS-GAIL perform relatively poorly, failing to capture the data distribution. These results represent the best training epoch out of the 5000 performed during training, as we observed that both baseline GAIL methods exhibit quite unstable training dynamics. Figure 6, which plots the joint velocity-occupancy JSD metric across the training epochs for a multi-agent evaluation of 4000 timesteps across 3 random seeds, shows that Horizon GAIL is noticeably more stable.

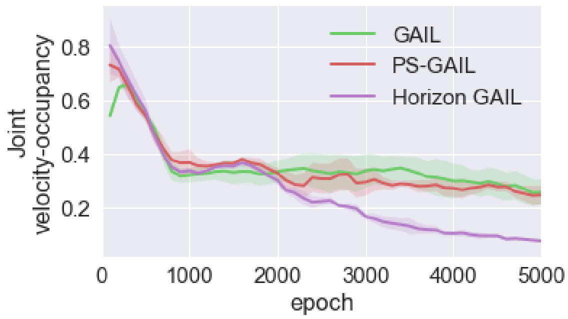

Fig. 6. Progression of Joint velocity-occupancy JSD metric through training, indicating difference in stability between our method (Horizon GAIL) and other GAIL baselines across 3 random seeds.

With respect to PS-GAIL, we observed that the curriculum parameter was relatively hard to tune. For example, adding agents too soon causes the discriminator to learn too quickly that these agents are not real.

Another notable observation is that BC performs well when compared to both baseline GAIL methods. This result can be attributed to the abundance of data available for training. From Figure 5 however we can see that the qualitative performance of these policies is relatively poor when compared to Horizon GAIL. As expected, the BC baseline quickly diverges from plausible trajectories, as minor errors compound over time. The long evaluation times exacerbate this effect. Horizon GAIL avoids compounding error problems associated with BC through interaction with the environment. It also avoids unstable training related with GAIL through the gradually increasing horizon. This yields stable, plausible trajectories with fewer collisions than any other method.[1]

VI. Conclusion

This paper presented a novel method for learning from demonstration in the wild that can leverage abundance of freely available videos of natural behaviour. In particular, we proposed ViBe, a new approach to learning models of road user behaviour from unlabelled raw video data of a traffic scene collected from a single, monocular, uncalibrated camera with ordinary resolution. ViBe calibrates the camera, detects relevant objects, tracks them reliably through time, and uses the resulting trajectories to learn driver policies via a novel LfD method. The learned policies are finally deployed in a simulation of the scene developed using the Unity game engine. According to several metrics our LfD method exhibits better and more stable learning than baselines such as GAIL and BC.

[1] Accompanying video: `https://youtu.be/3VK4tQTHeHc`. Supplementary material: `https://arxiv.org/abs/1811.03516`

References

[1] T. Zhang, *et al.*, "Deep imitation learning for complex manipulation tasks from virtual reality teleoperation," *arXiv preprint arXiv:1710.04615*, 2017.

[2] T. Osa, *et al.*, "An algorithmic perspective on imitation learning," *Foundations and Trends® in Robotics*, vol. 7, no. 1-2, pp. 1–179, 2018.

[3] B. D. Argall, *et al.*, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[4] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Advances in Neural Information Processing Systems*, 2016, pp. 4565–4573.

[5] K. He, *et al.*, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[6] R. Girshick, *et al.*, "Detectron," https://github.com/facebookresearch/detectron, 2018.

[7] R. Girshick, "Fast R-CNN," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2015.

[8] S. Ren, *et al.*, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Neural Information Processing Systems (NIPS)*, 2015.

[9] K. He, *et al.*, "Mask R-CNN," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017.

[10] W. Liu, *et al.*, "Ssd: Single shot multibox detector," in *ECCV*, 2016.

[11] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *CoRR*, vol. abs/1804.02767, 2018. [Online]. Available: http://arxiv.org/abs/1804.02767

[12] T. Lin, *et al.*, "Focal loss for dense object detection," in *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, 2017, pp. 2999–3007.

[13] N. Wojke, *et al.*, "Simple online and realtime tracking with a deep association metric," in *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2017, pp. 3645–3649.

[14] N. Wojke and A. Bewley, "Deep cosine metric learning for person re-identification," in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2018, pp. 748–756.

[15] L. Leal-Taixé, *et al.*, "Learning by tracking: Siamese cnn for robust target association," *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 418–425, 2016.

[16] L. Bertinetto, *et al.*, "Fully-convolutional siamese networks for object tracking," in *ECCV Workshops*, 2016.

[17] A. Bewley, *et al.*, "Simple online and realtime tracking," in *2016 IEEE International Conference on Image Processing (ICIP)*, 2016, pp. 3464–3468.

[18] E. Bochinski, *et al.*, "High-speed tracking-by-detection without using image information," in *International Workshop on Traffic and Street Surveillance for Safety and Security at IEEE AVSS 2017*, Lecce, Italy, Aug. 2017. [Online]. Available: http://elvera.nue.tu-berlin.de/files/1517Bochinski2017.pdf

[19] M.-C. Chang, *et al.*, "Video analytics in smart transportation for the aic18 challenge," in *CVPR Workshop (CVPRW) on the AI City Challenge*, 2018.

[20] A. Y. Ng, *et al.*, "Algorithms for inverse reinforcement learning." in *Icml*, 2000, pp. 663–670.

[21] B. D. Ziebart, *et al.*, "Maximum entropy inverse reinforcement learning." in *AAAI*, vol. 8. Chicago, IL, USA, 2008, pp. 1433–1438.

[22] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004, p. 1.

[23] Y. Duan, *et al.*, "One-shot imitation learning," in *Advances in neural information processing systems*, 2017, pp. 1087–1098.

[24] C. Finn, *et al.*, "Guided cost learning: Deep inverse optimal control via policy optimization," in *International Conference on Machine Learning*, 2016, pp. 49–58.

[25] Y. Liu, *et al.*, "Imitation from observation: Learning to imitate behaviors from raw video via context translation," *arXiv preprint arXiv:1707.03374*, 2017.

[26] E. Zhan, *et al.*, "Generative multi-agent behavioral cloning," *arXiv preprint arXiv:1803.07612*, 2018.

[27] B. D. Ziebart, *et al.*, "Navigate like a cabbie: Probabilistic reasoning from observed context-aware behavior," in *Proceedings of the 10th international conference on Ubiquitous computing*. ACM, 2008, pp. 322–331.

[28] X. B. Peng, *et al.*, "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills," *arXiv preprint arXiv:1804.02717*, 2018.

[29] D. Vasquez, *et al.*, "Inverse reinforcement learning algorithms and features for robot navigation in crowds: an experimental comparison," in *IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, 2014, pp. 1341–1346.

[30] Y. Li, *et al.*, "Infogail: Interpretable imitation learning from visual demonstrations," in *Advances in Neural Information Processing Systems*, 2017, pp. 3812–3822.

[31] N. Lee, *et al.*, "Desire: Distant future prediction in dynamic scenes with interacting agents," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 336–345.

[32] P. Henry, *et al.*, "Learning to navigate through crowded environments," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 981–986.

[33] A. Kuefler, *et al.*, "Imitating driver behavior with generative adversarial networks," in *Intelligent Vehicles Symposium (IV), 2017 IEEE*. IEEE, 2017, pp. 204–211.

[34] R. P. Bhattacharyya, *et al.*, "Multi-agent imitation learning for driving simulation," *arXiv preprint arXiv:1803.01044*, 2018.

[35] J. Colyar and J. Halkias, "Us highway 101 dataset," *Federal Highway Administration (FHWA), Tech. Rep. FHWA-HRT-07-030*, 2007.

[36] Y. Aytar, *et al.*, "Playing hard exploration games by watching youtube," *arXiv preprint arXiv:1805.11592*, 2018.

[37] X. B. Peng, *et al.*, "Sfv: Reinforcement learning of physical skills from videos," *arXiv preprint arXiv:1810.03599*, 2018.

[38] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proceedings of the tenth international conference on machine learning*, 1993, pp. 330–337.

[39] J. K. Gupta, *et al.*, "Cooperative multi-agent control using deep reinforcement learning," in *International Conference on Autonomous Agents and Multiagent Systems*. Springer, 2017, pp. 66–83.

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

[41] S. Ross, *et al.*, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 627–635.

[42] S. Ross and D. Bagnell, "Efficient reductions for imitation learning," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 661–668.

[43] I. Goodfellow, *et al.*, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[44] J. Schulman, *et al.*, "Trust region policy optimization," in *International Conference on Machine Learning*, 2015, pp. 1889–1897.

[45] J. Schulman *et al.*, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[46] T.-Y. Lin, *et al.*, "Microsoft coco: Common objects in context," in *k*, 2014.

[47] D. G. R. Bradski and A. Kaehler, *Learning Opencv, 1st Edition*, 1st ed. O'Reilly Media, Inc., 2008.

[48] Z. Wang, *et al.*, "Robust imitation of diverse behaviors," in *Advances in Neural Information Processing Systems*, 2017, pp. 5320–5329.

[49] J. Song, *et al.*, "Multi-agent generative adversarial imitation learning," *arXiv*, 2018.

[50] OpenAI, "Openai five," 2018. [Online]. Available: https://blog.openai.com/openai-five/

[51] E. Ristani, *et al.*, "Performance measures and a data set for multi-target, multi-camera tracking," in *ECCV Workshops*, 2016.

[52] K. Shiarlis, *et al.*, "Inverse reinforcement learning from failure," in *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, 2016, pp. 1060–1068.

Appendix

A. Vision

The vision pipeline discussed in Section IV-A calibrates the camera, detects relevant objects, tracks them from one frame to the next, and provides our LfD system with demonstrations. Here we provide further details on the tracking system.

Tracking is performed in two consecutive stages using appearance features and IOU (intersection-over-union) described in Section II. In the first stage, for a certain frame N in the video, given existing tracks and detections generated from Mask R-CNN [9], an appearance cost matrix is computed, where each element is the distance between the features of each detection and each track. The feature distances between one detection in the Nth frame and each detection in a track are calculated, and the minimum value is selected.

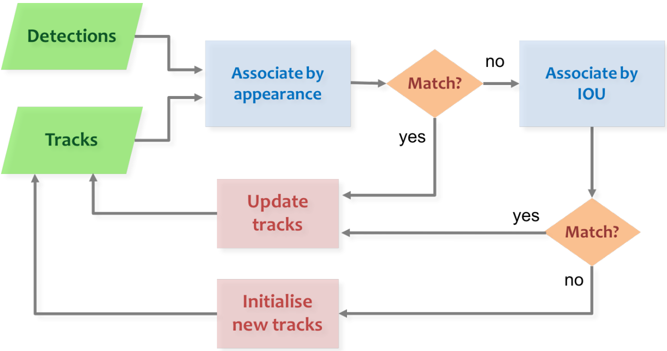

Fig. 7. Schematic of ViBe's tracker: Detections generated from Mask R-CNN are matched with tracks in two stages, using appearance features and IOU. Detections with no matches are initialised as new tracks.

For each track, the associated 3D Kalman filter predicts the next position of the track. We only consider detections for the next step to be valid if they fall within a fixed radius of the Kalman filter's prediction. Additionally, we also only consider detections to be valid if they have a cosine distance in the Siamese network's feature space below a certain threshold compared to the current track. These two gates are applied on the appearance cost matrix to generate a new matrix where each element outside the valid range is substituted by a large value. The Hungarian algorithm is then applied on this new cost matrix to provide matched tracks and detections.

The unmatched tracks and detections are then fed into the second stage. An IOU cost matrix is calculated between the unmatched detections and the last successful detection of unmatched tracks. Similarly, a threshold is applied to the appearance cost matrix between the unmatched detections and the unmatched tracks. The IOU matrix is modified according to the appearance gating. The Hungarian algorithm is applied and the result provides the matched detections and matched tracks. Unmatched detections are used to initialise new tracks. Unmatched tracks are removed from the tracking process if the number of frames without detection association exceeds a threshold. The tracks that have been matched in this frame given the 2 stages and the newly initialised tracks are then fed into the next frame, which repeats the above process.

Figure 7 depicts this process. The finalised tracks are then played back in our simulator to provide the demonstration trajectories used by our LfD algorithm (see Section IV-B).

B. LfD

Here we provide some implementation details about our LfD approach. Given the trajectories extracted by our visual processing module, ViBe uses the simulator to learn a policy that imitates those trajectories using our novel Horizon GAIL method, described in Section IV-C. Algorithm 1 provides a complete overview of our training scheme.

Algorithm 1 Horizon Curriculum for GAIL

```
Initialise policy $\pi_\theta$, discriminator $D_\phi$, expert demonstrations $\mathcal{D}_E$
for $h = 1 \dots T$ do
    Sample expert trajectory: $\tau_E \sim \mathcal{D}_E$
    for $t = 0, h, 2h, \dots, T - h$ do
        Use expert observation $s_t^E$ to initialise the agent and initialise the environment to its corresponding state at time $t$
        Sample an agent's trajectory of length $h$: $\tau \sim \pi_{\theta_i}$
    end for
    Sample $M$ observation-action pairs $\chi \sim \tau$ and $M$ pairs $\chi_E \sim \tau_E$
    Update the discriminator parameters from $\phi_i$ to $\phi_{i+1}$ with the gradient:
        $\mathbb{E}_{(s_m,a_m)\in\chi}[\nabla_\phi \log(D_\phi(s_m, a_m))] + \mathbb{E}_{(s_m^E,a_m^E)\in\chi_E}\left[\nabla_\phi \log(1 - D_\phi(s_m^E, a_m^E))\right]$
    Compute reward $\forall(s_m, a_m) \in \chi$ using the discriminator: $r_m = -\log(D_{\phi_{i+1}}(s_m, a_m))$
    Take a policy step from $\theta_i$ to $\theta_{i+1}$, with any policy optimisation method
end for
```

Horizon GAIL bootstraps learning from the expert's states, s_t^E, to ensure a reliable reward signal from the discriminator and helps stabilise learning by using a novel horizon curriculum that slowly increases the number of timesteps, horizon h, for which the agent interacts with the simulator.

Gradually moving from single step state-action pairs to more difficult multi-step trajectories allows the generator and discriminator to jointly learn to generalise to longer sequences of behaviour and match the expert data more closely while ensuring the discriminator does not collapse early in training.

The results of our LfD evaluation can be seen Section V-B. Here we extend Figure 6 to also include speed and occupancy JSD metrics (described in Section V-A) across the training epochs for a multi-agent evaluation of 4000 timesteps of the held-out data across three random seeds (see Figure 8). Throughout training, Horizon GAIL exhibits noticeably more stable behaviour across these metrics.

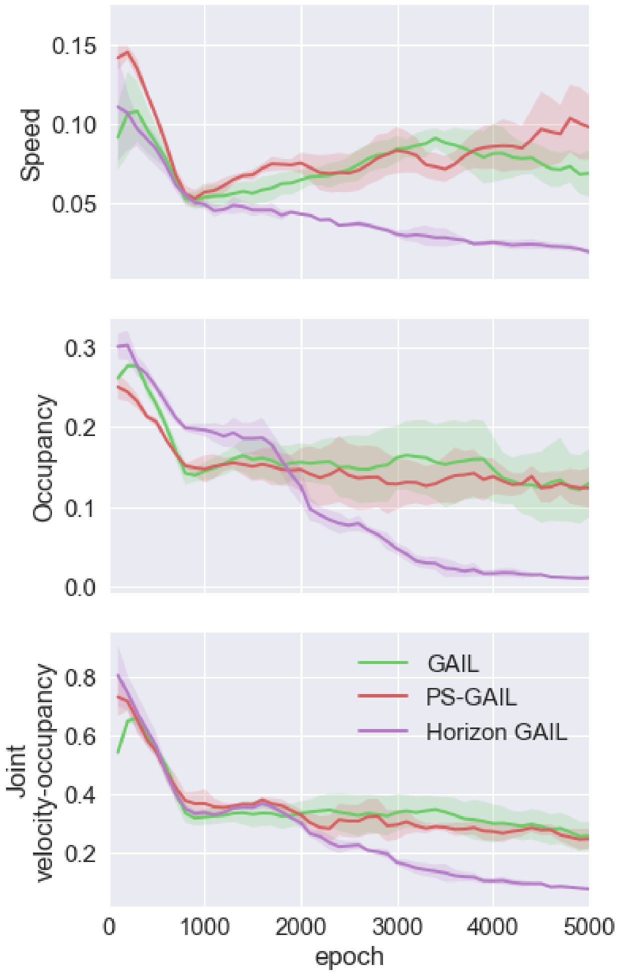

Fig. 8. Progression of speed, occupancy and joint velocity-occupancy JSD metrics throughout training for our method (Horizon GAIL) and other GAIL baselines across three random seeds.