Inverse Reinforcement Learning with Hybrid Weight Tuning for Autonomous Maneuvering

Yu Shen[1], Weizi Li[2], and Ming C. Lin[1]
https://gamma.umd.edu/researchdirections/autonomousdriving/eirl/

***Abstract*— Despite significant advancements, collision-free navigation in autonomous driving is still challenging. On one hand, the perception module needs to interpret multimodal unstructured data and produce structured data of the environment. On the other hand, the navigation module needs to balance the use of machine learning and motion planning in order to achieve efficient and effective control of the vehicle. We propose a novel framework combining context-aware multi-sensor perception and inverse reinforcement learning with hybrid weight tuning (IRL-HWT) for autonomous maneuver. IRL-HWT incorporates several attributes including non-uniform prior for features, hybrid weight tuning based on trust-region optimization, parameter reuse for continuous training, and learning from accidents. These attributes help reduce the number of collisions of the vehicle up to 41%, increase the training efficiency by 2.5x, and obtain higher test scores up to two orders of magnitude. Overall, our method can enable the vehicle to drive 10x further than other methods, while achieving collision avoidance over both static and dynamic obstacles.**

I. Introduction

Autonomous vehicles (AVs) have the potential to contribute to a more efficient and safer transportation system by alleviating traffic congestion and reducing the number of accidents. In general, autonomous driving can be realized via either an end-to-end approach or a mediated-perception approach [1]. An end-to-end approach meaning that raw sensor data are directly mapped to control commands (e.g., steering angles), which usually results in a succinct training pipeline at the cost of model interpretability. A mediated-perception approach, on the other hand, decouples perception and navigation, thus offering better model interpretability and enhanced driving safety. However, a mediated-perception approach commonly adopts a planning algorithm for navigating an AV—a process that can be computationally expensive, given the requirement of holistic environment information for achieving global optimality and planning in high-dimensional state space.

We propose a novel mediated-perception approach for autonomous driving. Our approach consists of context-aware multi-sensor perception and inverse reinforcement learning with hybrid weight tuning mechanism (IRL-HWT). The perception module interprets unstructured information (i.e., images and point clouds) of an environment using multiple sensors with different viewpoints, extract context-aware information, and produces structured information, such as the size and position of an object in the environment. IRL-HWT then takes the structured information along with expert trajectories as input to learn a control policy for autonomous driving. In particular, IRL-HWT addresses a fundamental limitation of the original IRL, which imposes a *uniform prior* on all features. This limitation can lead to undesired behaviors (e.g., frequent collisions) even when the feature expectation of the learned policy closely matches the feature expectation of the expert policy (see Sec. IV for a concrete example). IRL-HWT solves this problem by incorporating a non-uniform prior for features and update the weights of features using an automatic hybrid weight tuning mechanism that is based on trust-region optimization. The pipeline of our approach is illustrated in Fig. 1.

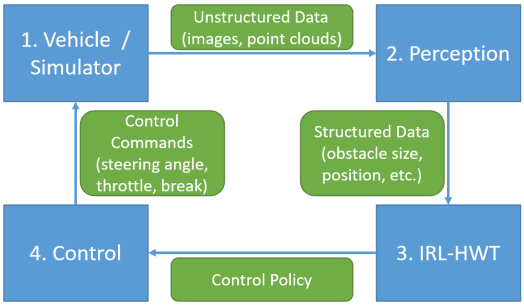

Fig. 1. System pipeline. At each time step, the vehicle/simulator generates unstructured data such as images and point clouds. These data are processed by the perception module to produce structured data, which are then used by the IRL-HWT module to learn a control policy for autonomous driving.

Our approach has several advantages. First of all, as a mediated-perception approach, instead of using a planning algorithm for navigating an AV at all time steps, we only use a planning algorithm to generate expert demonstrations for IRL-HWT to imitate. Once learned, the control policy can operate in real time with a small number of features. This design choice greatly reduces the computational overhead of using a planning algorithm. Second, IRL-HWT complements the original IRL [2] with the flexibility to impose a non-uniform prior on important features for a specific problem, for example, collision for autonomous driving. Third, the hybrid weight tuning scheme allows the learned policy to adapt to different environments while maintaining desired behaviors and avoiding aggressive exploratory behaviors. Fourth, since our approach is based on reinforcement learning, compared to using supervised learning for imitation, IRL-HWT is more robust in rare situations and can generalize better in new environments [2]. Lastly, similar to

[1]Yu Shen and Ming C. Lin are with the Department of Computer Science, University of Maryland at College Park {yushen,lin}@cs.umd.edu
[2]Weizi Li is with the Department of Computer Science, University of Memphis wli@memphis.edu

ADAPS [3], alternative safe trajectories are generated during the analysis of an accident. This approach enables the learning algorithm to *learn from accidents*, which is crucial as collecting accident data from the real world is impractical.

The effectiveness and efficiency of our approach are demonstrated in a variety of experiments. To show IRL-HWT can work beyond perfect perception, i.e., use ground-truth data from the environment, we run IRL-HWT in a complete autonomous system as described in Sec. III-A. Overall, our method can enable the AV to drive safely 10x further than the other methods. The attribute non-uniform prior can assist in reducing the number of collisions of the AV up to 41%; the use of learned model parameters for continuous training can result in 2.5x faster training; and learning from accidents can help achieve higher test scores up to two orders of magnitude. In addition to quantitative results, we qualitatively show that our method can steer the AV to avoid both static and dynamic obstacles, even in the presence of a narrow passage (see supplementary video).

II. Related Work

In this section, we briefly discuss the related work of different aspects of our framework.

A. Autonomous Driving

Various methods have been proposed to solve the perception, planning, and control problems for autonomous driving [4], [5]. Examples of the end-to-end approach include end-to-end reinforcement learning [6] and end-to-end imitation learning [3], [7], [8], [9]. These approaches usually require a large amount of training data in order to be robust in rare cases, such as pre-accident scenarios. In addition, the use of deep neural networks in these approaches to directly map raw sensor data to control commands can lead to low model interpretability.

Examples of the mediated-perception approach include perception plus motion planning [10] and perception plus learning-based planning [7]. Because of the decomposition of perception and navigation, these approaches enjoy better model interpretability, hence improved driving safety. Recently, Li et al. propose ADAPS [3], an end-to-end imitation learning framework that enables an AV to learn from accidents. Compared to ADAPS, our work proposes a new architecture that can not only fuse data from different modalities but also generalize better, as it relies on reinforcement learning rather than using supervised learning to imitate the expert's behaviors.

B. Inverse Reinforcement Learning (IRL)

As an effective technique for imitation learning, IRL involves two steps: 1) learning a reward function from experts' demonstrations and 2) using the acquired reward function for reinforcement learning to learn a control policy [2]. To provide some examples, Sharifzadeh et al. [11] apply Deep Q-Networks to extract a reward function in large state space. You et al. [12] use deep neural networks to approximate the latent reward function of the expert and then apply deep Q-learning to obtain the control policy.

Compared to the original IRL [2], our approach allows the use of a non-uniform prior, instead of a uniform prior, on the features. This can prevent important features from receiving trivial weights during the learning process. In addition, our approach uses the parameters of a pre-trained policy as the start point for continuous training, which can improve training efficiency while maintaining the model performance.

III. Approach

A. Framework Overview

Our architecture combines context-aware multi-sensor perception and inverse reinforcement learning via hybrid weight tuning (IRL-HWT). The perception module takes multiple sensors' data as input and produces structured data, which are then used by IRL-HWT to learn a control policy.

We assume that the AV can obtain a global map from an external service, and compute its position and rotation with on-board GPS and Inertial Measurement Unit (IMU). We further assume that the AV knows beforehand a few sparse waypoints on its path to the goal, and the task of the AV is converted to reach the waypoints consecutively.

Our perception module can produce semantic-rich structured data to learn a reward function by utilizing the context-aware semantic information. Features with clear semantic interpretations can help construct non-linear features such as "whether there is a car in front within 3 meters", resulting in more flexible decision-making. This is particularly useful considering that IRL restricts the reward function to be a linear combination of the features.

The IRL-HWT training process, shown in Fig. 2, consists of an offline step and an online step. In the offline step, we use a planning algorithm as the expert to generate driving trajectories and compute the expert's feature expectation. In the online step, we learn the feature weights under a *non-uniform prior* given the expert's policy and the learned policy at the current iteration. Then, we construct a reward function using the learned feature weights. By further adopting the notion of *learning from accidents* [3], we use the resulting additional training data along with the newly constructed reward function for RL to update the learned policy. Inspired by transfer learning [13], the model parameters from the previous iteration are used as a starting point in the updating process.

B. Context-aware Multi-sensor 3D Perception

We simulate three RGB cameras for the front-view, left-view, and right-view, respectively, as well as a 360-degree Lidar of the AV. We combine one RGB image and the point cloud on each side to detect nearby vehicles. Then, we merge the results from all sides to obtain an overall view. The module works as follows.

First, it generates the front-view data, which contain segmentation and depth maps, and bird's-eye-view image from the point cloud. By having the calibration data between the Lidar and camera, we can then align the front-view data with the RGB images. In the second step, we combine and feed the aligned front-view data with the RGB images into

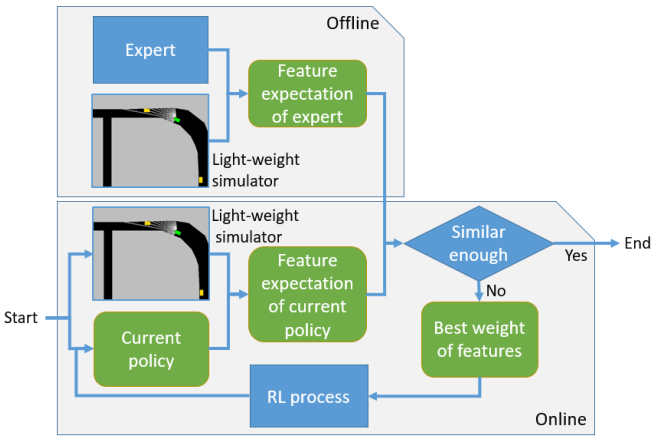

Fig. 2. IRL-HWT training process. We compute the expert's trajectories and feature expectation offline. We then use the learned reward function to compute the feature expectation of a learned policy online. If the feature expectations of the expert and the learned policy differ too much, RL is used to update the learned policy.

the feature extractor to obtain the front-view feature map. We apply the same procedure for the bird's-eye-view image to obtain the bird's-eye-view feature map. Then, we use the region proposal network (RPN) [14] and detection network from AVOD [15] to obtain the perception results for the front-view. Similarly, we obtain the perception results for the left-view and right-view. Finally, we merge all perception results together using the extrinsic calibration data of the three cameras.

The output of the perception module contains 3D bounding boxes and types of nearby dynamic obstacles. Our approach can also be extended to detect static obstacles when needed. The context-aware, multi-sensor 3D perception detects different types of information to produce features used by IRL. For example, lidar can detect depth information to ensure safety while camera can identify object types to plan actions.

C. Inverse Reinforcement Learning Via Hybrid Weight Tuning (IRL-HWT)

The original IRL achieves imitation learning by first computing the expert's feature expectation $\hat{\mu}(\pi_E) = \frac{1}{m}\sum_{i=1}^{m}\sum_{t=0}^{\infty}\gamma^t\phi(s_t^{(i)})$, given m trajectories $\{s_0^{(i)}, s_1^{(i)}, \dots\}_{i=1}^{m}$ from the expert's policy π_E, the discount factor γ, and the feature vector $\phi(\cdot)$. Then, IRL learns w ($w \in \mathbb{R}^k$ and $\|w\|_1 \leq 1$) given π at the current iteration and π_E, and synthesizes a reward function $R(s) = w \cdot \phi(s)$, where s is the state of the environment. Next, the policy π is re-learned using RL. This iterative process continues until $\|\mu(\pi) - \hat{\mu}(\pi_E)\|_2 \leq \epsilon$. The final policy is selected among all learned policies from all iterations.

The features used by IRL-HWT are based on structured data from the perception module, which include bounding boxes of nearby vehicles and static obstacles in the scene. See detailed feature list in Sec. IV.

One fundamental limitation of IRL is that it imposes a *uniform prior* on all features, causing small weights to be possibly assigned to some crucial features during the learning process. For example, in the context of driving, we find that the feature *collision* can receive a small weight as a result of any collision behavior of the AV will terminate a training episode. This limitation of IRL can lead to subpar performance of the task *learning to drive* (see Sec. IV-B). To give an example, the expert can drive the car safely (without any collision), while a randomly initialized policy can hardly drive the car far without collision. In this case, the feature expectation of the expert is calculated using extended, "global" trajectories, while the feature expectation of the learned policy is calculated using short, "local" trajectories. This discrepancy is likely to cause some important features to receive small weights during the minimization process of the feature expectations in IRL. A concrete example of this phenomenon from our experiments is shown in Table I.

TABLE I

LIMITATION OF IRL. IRL LEARNS FEATURE WEIGHTS w BY MINIMIZING THE DIFFERENCE OF FEATURE EXPECTATION BETWEEN THE EXPERT'S POLICY $\mu(\pi_E)$ AND A RANDOM POLICY $\mu(\pi_0)$. WHILE BOTH POLICIES HAVE SMALL EXPECTATIONS FOR THE FEATURE *collision*, THE ACTUAL TRAJECTORY FROM THE EXPERT'S POLICY CAN BE MUCH LONGER THAN THE TRAJECTORY FROM A RANDOM POLICY. AS A RESULT, IRL ASSIGNS A NEGLIGIBLE WEIGHT TO *collision* (I.E., $-5.34e-06$).

feature	left dist.	front dist.	right dist.	...	**collision**
weight	0.113	0.184	−0.124	...	-0.00000534
$\mu(\pi_E)$	139.14	369.50	25.84	...	0
$\mu(\pi_0)$	40.31	216.06	142.87	...	0.005

To alleviate the aforementioned limitation of IRL, we propose IRL-HWT—an approach that not only incorporates a *non-uniform prior* on the features by allowing users to specify the weights of certain features for ensuring essential properties of a task, e.g., collision for driving, but also uses a hybrid weight tuning to update the policy so that the AV can adapt to different environments while maintaining desired behaviors. Formally, the overall weights $w = [w_m, w_l]$ consist of empirically-initialized weights $w_m = [w_1, \dots, w_k]$ and the weights to be learned from scratch $w_l = [w_{k+1}, w_{k+2}, \dots, w_n]$, where n is the total number of features. To achieve an optimal policy in diverse environments, we use trust-region optimization [16] to automatically tune w_m by avoiding violent exploratory behaviors. Specifically, in the ith iteration, we first try to update ϵ and w by solving the following quasi-convex optimization program:

$$\epsilon^{(i)} = \max_{\substack{w^{(i+1)}:\|w^{(i+1)}\|_2\leq 1 \\ \|w_m^{(i+1)} - w_m^{(i)}\|_2 \leq \Delta}} \{\min_{j\in\{0,\dots,i\}} (w^{(i+1)})^T(\mu(\pi_E) - \mu(\pi^{(j)}))\}, \tag{1}$$

where Δ is the trust-region radius ($\Delta = \Delta^{(i)}$). The ratio $\epsilon^{(i)}/\epsilon^{(i-1)}$ is used to determine the acceptance of the newly updated w. Since the predicted upper bound of $\epsilon^{(i)}/\epsilon^{(i-1)}$ is $\frac{n}{\sqrt{n^2+(1-\gamma)^2\epsilon^2}}$ [2], we set the acceptance condition to

$$\frac{\epsilon^{(i)}}{\epsilon^{(i-1)}} \leq \alpha \frac{n}{\sqrt{n^2 + (1-\gamma)^2 \epsilon^2}}, \tag{2}$$

where $\alpha < 1$ is the coefficient parameter. If the condition is failed, we drop the newly found w, and update ϵ and w by solving Eq. 1 with $\Delta = 0$. As the last step of one iteration, we update Δ as follows:

$$\Delta^{(i+1)} = \begin{cases} \min(c_u \Delta^{(i)}, b_u) & \text{Eq. 2 met} \\ \max(c_l \Delta^{(i)}, b_l) & \text{Eq. 2 failed p times} \\ \Delta^{(i)} & \text{otherwise} \end{cases} \tag{3}$$

where b_u and b_l are the upper- and lower-bound of the trust-region radius, and $c_u > 1$ and $c_l < 1$ are the coefficients.

Additionally, IRL retrains π at each iteration, such a process can be inefficient. In IRL-HWT, we improve the training efficiency by using the learned model parameters $\theta^{(i)}$ of $\pi^{(i)}$ at the ith iteration as the initial parameters for training $\pi^{(i+1)}$ at the $(i+1)$th iteration. The whole learning process stops when $\|\mu(\pi) - \hat{\mu}(\pi_E)\|_2 < \epsilon$. The complete IRL-HWT algorithm is shown in Algorithm 1.

Algorithm 1: Inverse Reinforcement Learning Via Hybrid Weight Tuning (IRL-HWT)

Result: policy $\pi^{(i)}$
Initialization: Calculate $\mu(\pi_E)$ with expert trajectories;
Set $i = 0$, set $\epsilon, \gamma, \alpha, b_u, b_l, c_u, c_l, p$;
Randomly set the model parameters $\theta^{(0)}$ for $\pi^{(0)}$;
Compute $\mu(\pi^{(0)})$;
Set $w_m^{(0)}$ such that $\|w_m^{(0)}\|_2 < 1$;
Compute $\epsilon^{(0)} = \max_{w_l : \|w\|_2 \leq 1} w^T(\mu(\pi_E) - \mu(\pi^{(0)}))$, where $w = [w_m^{(0)} w_l]$;
Let $w_l^{(0)}$ store the above maximum value and set $w^{(1)} = [w_m \; w_l^{(1)}]$;
Set $\Delta^{(1)}$;
while $\epsilon^{(i)} > \epsilon$ **do**
  Set $i = i + 1$;
  Compute the reward function $R = ((w^{(i)})^T \phi)$;
  Using R and $\theta^{(i-1)}$ in RL to compute an optimal policy $\pi^{(i)}$;
  Compute $\mu(\pi^{(i)})$;
  Solve Optimization 1 with $\Delta = \Delta^{(i)}$, and get solution $\epsilon^{(i)}$ at $w^{(i+1)}$;
  if Condition 2 is True **then**
    Accept;
  else
    Reject, solve Optimization 1 with $\Delta = 0$, and update $\epsilon^{(i)}$ and $w^{(i+1)}$;
  end
  Set $\Delta^{(i+1)}$ with Equation 3;
end

Inspired by ADAPS [3], we adopt *learning from accident* to improve the training efficiency. Specifically, when the car crashes during the online training process, we will backtrack for certain frames and let the expert drive for certain frames to avoid collision. Both the crash and collision-free data will be (re-)used to train the policy. Essentially, by imposing a non-uniform prior on extracted features, we can introduce certain expert experience to the learning process directly. In addition, although the synthesized reward function is different at each iteration, they may embed useful information to share with each other. Thus, inspired by transfer learning [13], we reuse learned parameters for continuous training for faster convergence. Lastly, the process of *learning from accident* can result in both negative examples and positive examples in learning, leading to better performance and faster convergence.

D. Autonomous Driving Training Platform

Our platform is developed using the Unity game engine [17] and consists of the following components.

- Heavy-weight simulator for providing realistic 3D scenes and light-weight simulator for improving planning efficiency.
- Virtual sensors. Our platform supports simulated RGB camera, depth camera, lidar, gyroscope, and GPS.
- Portable learning algorithms. Our platform supports end-to-end learning, perception plus traditional planning, and perception plus learning-based planning. These algorithms can be switched and replaced at ease.
- Communication and control module. Our platform supports cross-platform communication with sockets. This feature enables remote control of simulated AVs using input from either a planning algorithm or user.
- Data collection module. Our platform can operate various types of data, including sensor data, calibration data, and environment data in any format. For example, our platform can generate data that are compatible with the KITTI dataset [18] for 3D object detection.

Compared to other platforms such as CARLA [19], ours has both heavy-weight and light-weight simulators, providing better flexibility and efficiency. To be concrete, while the perception module requires a realistic 3D scene, the planning module only needs information of the environment state such as distance to obstacles. Our platform enjoys decoupling the two simulators at different level of detail and avoids the prohibitive computational cost of updating the policy solely in the 3D environment. Using our platform, one can train the perception module in the heavy-weight simulator and the planning module in the light-weight simulator, and then combine the results for a system-level test.

IV. Experiments and Results

In this section, we detail our experiments and results. All experiments are conducted using an Intel(R) Xeon(TM) W-2123 CPU, an Nvidia GTX 1080 GPU, and 32G RAM.

A. Overall Performance

We compare IRL-HWT to the original IRL, and end-to-end imitation learning (IM). We choose IM as a method for comparison since it also learns from the expert trajectories. We use the method by Bojarski et al. [20] as the IM, and deep Q-learning with $[164, 150]$ hidden units in each of the 2 dense layers and 20% dropout rate in all RL-based methods (We also experimented with more complex network architectures up to 6 hidden layers, but found [164,150] with 2 hidden layers works best). The features used in both the original IRL and IRL-HWT are listed in Fig. 3.

ID	Feature name	Description
1	Waypoint reaching flag	1 if a waypoint is reached 0 otherwise
2	Distance reduction to waypoint	Distance (meter) reduction to waypoint compared to the previous step
3	Direction away from waypoint	Relative angle (radian) between the forward direction of AV and direction to waypoint
4	Collision flag	1 if collide 0 otherwise
5~25	Depth list	Depth (meter) in 21 directions uniformly distributed in 120 degree ([-60, 60]) of the forward direction in bird's-eye-view
26~46	Object type list	Object type of the first object that stops the ray from AV, in 21 directions uniformly distributed in 120 degree ([-60, 60]) of the forward direction in bird's-eye-view 0 if nothing is detected or unknown object 1 if static object 2 if dynamic object

Fig. 3. Features used in IRL and IRL-HWT (46 in total).

The action space contains 25 discrete (rational speed, acceleration) action pairs: 5 levels of rotational speed (steering angle) and 5 levels of acceleration (throttle value and break value). We set policy similarity threshold $\epsilon = 0.1$ and discount factor $\gamma = 0.99$ for both IRL and IRL-HWT. We collect 20,000 steps of the expert trajectory and train all methods for 24 hours (we also observe that training 12 more hours will not improve the performance). For IRL-HWT, specifically, we set the trust-region acceptance coefficient $\alpha = 0.9$, trust-region increasing and decreasing coefficients $c_u = 1.1, c_l = 0.9$, upper- and lower-bound $b_u = 0.05, b_l = 0.001$, and consecutive rejection number $p = 5$. We empirically initialize weights for the first 4 features in Fig. 3 with values [0.4, 0.01, -0.1, -0.8], and let IRL-HWT to learn the weights of depth- and object type-related features. The perception network is trained using 10,000 frames of simulated data from our platform.

Our evaluation criterion is *how far can the AV travel under a fixed number of steps*. The AV will stop if it finishes $1,000$ steps or is in collision. Since the AV is assumed to know beforehand a series of waypoints on its path to the goal, we compute the score based on the number of waypoints reached by the AV:

$$s_{final} = (n_{reached} + (1 - \frac{dist_{next}}{dist_{last\_next}})) \times s_{unit}, \quad (4)$$

where s_{final} is the final score, $n_{reached}$ is the number of waypoints reached, $dist_{next}$ is the distance between the last position on the car's trajectory and the next waypoint on the car's route, $dist_{last\_next}$ is the distance between the last reached waypoint to the next waypoint, and s_{unit} is the unit score by reaching a waypoint, which is set to 100. Note that a negative score may appear if $dist_{next} > dist_{last\_next}$, which happens when the car drives away from the next waypoint. We use three scenes shown in Fig. 4 for evaluation:

- Scene 1: open space with only moving vehicles;
- Scene 2: city street with only static obstacles;
- Scene 3: city street with moving vehicles and static obstacles.

Fig. 4. Screenshots of the three scenes used in our evaluation. From left to right: Scene 1, Scene 2, Scene 3.

We record the final scores $s_{final,2}$ and $s_{final,3}$ from Scene 2 and 3; and average trajectory length $l_{final,1}$, $l_{final,2}$, $l_{final,3}$ from Scene 1, 2, and 3. Because s_{final} is computed based on the waypoint position and there is no explicit waypoint in Scene 1, we do not compute $s_{final,1}$. All data are obtained by running each learning method for 100 times with randomly initialized scene configurations such as the start position and direction of the AV, and the start position, direction and speed of obstacle vehicles. Our method achieves the highest scores and can enable the AV to drive safely 10x further (and longer) than the other methods. The full results are shown in Table II.

TABLE II

OVERALL PERFORMANCE OF IRL-HWT VS. OTHER METHODS, USING THE SCORE DEFINED IN EQN. 4 AND SAFE-TRAJECTORY LENGTH. OUR METHOD NOT ONLY ACHIEVES THE *highest scores* BUT ALSO ENABLES THE AV TO DRIVE SAFELY *10x* FURTHER THAN THE OTHER METHODS.

Method	$s_{final,2}$	$s_{final,3}$	$l_{final,1}$	$l_{final,2}$	$l_{final,3}$
IM	77.4	60.1	105.6 m	53.7 m	44.7 m
IRL	110.7	59.7	228.8 m	69.4 m	33.2 m
Ours	**205.8**	**177.3**	**276.3** m	**748.4** m	**324.2** m

B. Attribute Effectiveness

The first attribute of our approach is the non-uniform prior for features. To test this attribute, we empirically set the weight for *collision* to be -0.8. We count the number of collisions from the original IRL and IRL+non-uniform prior (EIRL) by running both approaches for 10,000 steps. As shown in Table III, EIRL can reduce the number of collisions up to 41%.

The second attribute of our approach is *learned model parameters for continuous training*, which aims to improve training efficiency. From the results shown in Fig. 5, we can see that by having this attribute, we can achieve comparable model performance at *2.5x faster*.

TABLE III

THE NUMBER OF COLLISIONS IN DIFFERENT SCENES WITHIN 10,000 STEPS: ORIGINAL IRL VERSUS EIRL (IRL+NON-UNIFORM PRIOR). OUR EIRL CAN REDUCE THE NUMBER OF COLLISIONS UP TO *41%*.

Model	Scene 1	Scene 2	Scene 3
IRL	35	58	111
IRL+non-uniform prior	**33**	**41**	**93**

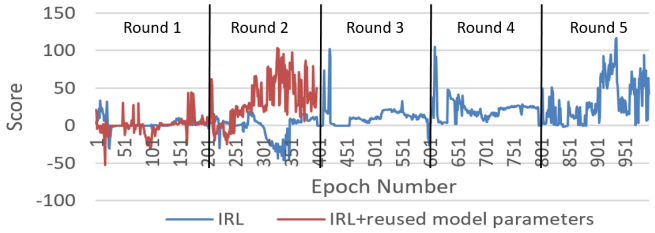

Fig. 5. By using previous model parameters for continuous training, we can achieve comparable score (model performance) 2.5x faster.

The last attribute of our approach is *learning from accidents*. We use ORCA [21] as the expert algorithm to generate alternative safe trajectories during the analysis of an accident. These trajectories are then used to generate additional training data for our algorithm. In Fig. 6, we show that the learning algorithm with this attribute added can achieve much higher scores up to *two orders of magnitude* in near collision scenarios under the same number of epochs.

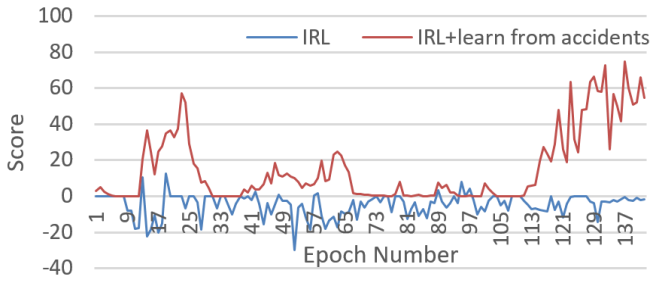

Fig. 6. Having the additional training data by *learning from accidents*, the learning algorithm achieves higher scores up to *two orders of magnitude* in near collision scenarios under the same number of epochs.

C. Driving Cases

Our method can enable safe autonomous driving (without collision) in scenes with both static and dynamic obstacles. We show the results of certain driving cases achieved by our method in Fig. 7. The full demo video can be found in the supplementary material.

In Fig. 7(a), we show that our method can steer the AV to make a left turn around the static obstacle while maintaining safe driving; In Fig. 7(b), we show that our method can lead the AV to avoid both static and dynamic obstacles. In particular, the AV (in green) steers to the right to avoid another vehicle coming from the opposite direction while moving away from the static obstacle; Lastly, we show that our approach can avoid multiple dynamic obstacles in Fig. 7(c). In this case, the AV (in green) is able to turn left to pass narrow space between two other vehicles.

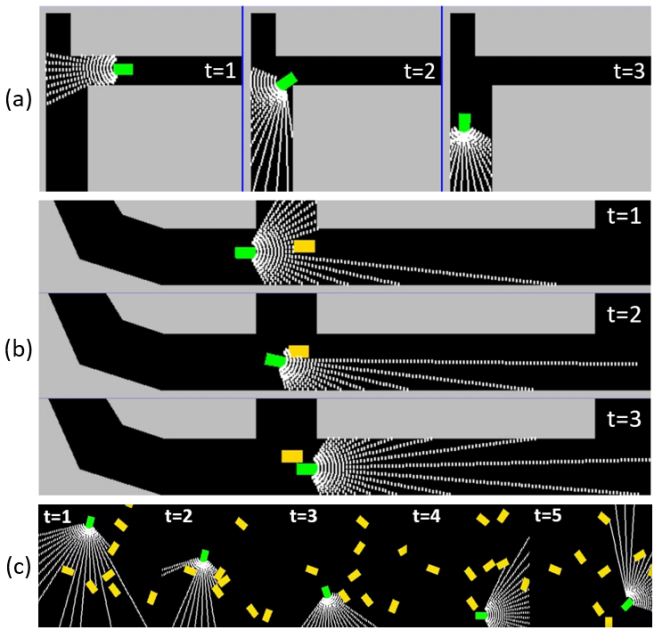

Fig. 7. (a) Static obstacle avoidance. Our method can lead the AV to make a left turn to avoid a static obstacle while maintaining safe driving. (b) Static and dynamic obstacle avoidance. The AV (in green) can avoid another vehicle (in yellow) coming from the opposite direction while steer away from the static obstacle. (c) Collision avoidance with multiple dynamic obstacles. Our method can direct the AV (in green) to avoid all nearby vehicles even when a narrow passage is presented.

D. High-level comparison

IM does not generalize well to new environments since it only mimics the expert trajectory. IRL can recover the reward function used by the expert but suffer from the limitation of a uniform prior. Our method addresses the limitations of IM and IRL by combining expert trajectory and domain knowledge. The hybrid weight tuning mechanism further enables our method to adapt to different environments and achieve desired behaviors.

V. CONCLUSION AND FUTURE WORK

We propose a framework integrating context-aware multi-sensor perception and inverse reinforcement learning via hybrid weight tuning (IRL-HWT) for autonomous driving. We evaluate our approach using a variety of experiments, over the entire algorithm and each individual component. As shown in all comparison results, our method outperforms the other state-of-the-art methods on all experiments.

There are some limitations of this work. First, the inference efficiency of the perception module can be improved. Second, our approach inherits other limitations of IRL, for example, information loss through encoding expert trajectories into a single feature expectation.

Future directions of this work are abundant. We plan to develop a more efficient perception module by leveraging the sparsity embedded in the input data [22]. We also hope to alleviate the information loss during the expert's feature expectation computation. Lastly, We plan to test our approach in dense virtual traffic [23] that is estimated and reconstructed using real-world traffic data [24], [25], [26].

REFERENCES

[1] S. Ullman, "Against direct perception," *Behavioral and Brain Sciences*, vol. 3, no. 3, pp. 373–381, 1980.

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

[3] W. Li, D. Wolinski, and M. C. Lin, "ADAPS: Autonomous driving via principled simulations," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019, pp. 7625–7631.

[4] W. Schwarting, J. Alonso-Mora, and D. Rus, "Planning and decision-making for autonomous vehicles," *Annual Review of Control, Robotics, and Autonomous Systems*, 2018.

[5] Q. Chao, H. Bi, W. Li, T. Mao, Z. Wang, M. C. Lin, and Z. Deng, "A survey on visual traffic simulation: Models, evaluations, and applications in autonomous driving," *Computer Graphics Forum*, vol. 39, no. 1, pp. 287–308, 2019.

[6] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani, "Deep reinforcement learning framework for autonomous driving," *Electronic Imaging*, vol. 2017, no. 19, pp. 70–76, 2017.

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

[8] M. Pfeiffer, M. Schaeuble, J. Nieto, R. Siegwart, and C. Cadena, "From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots," in *2017 ieee international conference on robotics and automation (icra)*. IEEE, 2017, pp. 1527–1533.

[9] F. Codevilla, M. Miiller, A. López, V. Koltun, and A. Dosovitskiy, "End-to-end driving via conditional imitation learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–9.

[10] M. Werling, S. Kammel, J. Ziegler, and L. Gröll, "Optimal trajectories for time-critical street scenarios using discretized terminal manifolds," *The International Journal of Robotics Research*, vol. 31, no. 3, pp. 346–359, 2012.

[11] S. Sharifzadeh, I. Chiotellis, R. Triebel, and D. Cremers, "Learning to drive using inverse reinforcement learning and deep q-networks," *arXiv preprint arXiv:1612.03653*, 2016.

[12] C. You, J. Lu, D. Filev, and P. Tsiotras, "Advanced planning for autonomous vehicles using reinforcement learning and deep inverse reinforcement learning," *Robotics and Autonomous Systems*, vol. 114, pp. 1–18, 2019.

[13] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, 2020.

[14] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[15] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander, "Joint 3d proposal generation and object detection from view aggregation," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1–8.

[16] A. R. Conn, K. Scheinberg, and L. N. Vicente, *Introduction to derivative-free optimization*. SIAM, 2009.

[17] Unity, "Unity game engine," https://unity.com/, 2020.

[18] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 3354–3361.

[19] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[20] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[21] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," pp. 3–19, 2011.

[22] L. Lin, W. Li, and S. Peeta, "Efficient data collection and accurate travel time estimation in a connected vehicle environment via real-time compressive sensing," *Journal of Big Data Analytics in Transportation*, vol. 1, no. 2, pp. 95–107, 2019.

[23] D. Wilkie, J. Sewall, W. Li, and M. C. Lin, "Virtualized traffic at metropolitan scales," *Frontiers in Robotics and AI*, vol. 2, p. 11, 2015.

[24] W. Li, D. Nie, D. Wilkie, and M. C. Lin, "Citywide estimation of traffic dynamics via sparse GPS traces," *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 3, pp. 100–113, 2017.

[25] W. Li, D. Wolinski, and M. C. Lin, "City-scale traffic animation using statistical learning and metamodel-based optimization," *ACM Trans. Graph.*, vol. 36, no. 6, pp. 200:1–200:12, Nov. 2017.

[26] W. Li, M. Jiang, Y. Chen, and M. C. Lin, "Estimating urban traffic states using iterative refinement and wardrop equilibria," *IET Intelligent Transport Systems*, vol. 12, no. 8, pp. 875–883, 2018.