Visual-based Autonomous Driving Deployment from a Stochastic and Uncertainty-aware Perspective

Lei Tai Peng Yun Yuying Chen Congcong Liu Haoyang Ye Ming Liu

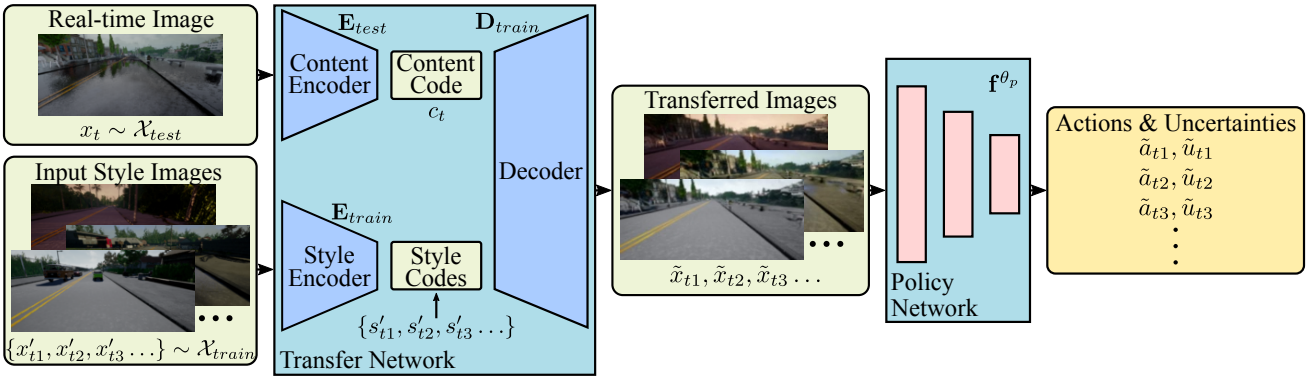


Fig. 1: The proposed end-to-end visual navigation deployment pipeline. When applying the end-to-end visual-based driving policy in real-time, an image from the mounted sensor is first translated to the training domain under various conditions through a stochastic generator. An uncertainty-aware imitation learning policy forward processes all the transferred images to output actions with uncertainties. The uncertainty can be used to refer to the most certain action.

***Abstract*— End-to-end visual-based imitation learning has been widely applied in autonomous driving. When deploying the trained visual-based driving policy, a deterministic command is usually directly applied without considering the uncertainty of the input data. Such kind of policies may bring dramatical damage when applied in the real world. In this paper, we follow the recent *real-to-sim* pipeline by translating the testing world image back to the training domain when using the trained policy. In the translating process, a stochastic generator is used to generate various images stylized under the training domain randomly or directionally. Based on those translated images, the trained uncertainty-aware imitation learning policy would output both the predicted action and the data uncertainty motivated by the *aleatoric* loss function. Through the uncertainty-aware imitation learning policy, we can easily choose the safest one with the lowest uncertainty among the generated images. Experiments in the *Carla* navigation benchmark show that our strategy outperforms previous methods, especially in dynamic environments.**

I. Introduction

End-to-end visual-based driving has received various interest from both the deep reinforcement learning [1], [2] and imitation learning [3] perspectives. In this paper, we mainly consider visual-based imitation learning, where a model is trained to guide a vehicle behaving similarly to a human demonstrator based on visual information. As a model-free method, raw visual information and other related measurements are taken as the input of a deep model, which is commonly a deep convolutional neural network (CNN) model. The deep model then outputs control commands directly, like steering and acceleration. It has been successfully applied in both indoor navigation [4] and outdoor autonomous driving [3].

Though learning-based methods have achieved many breakthroughs for autonomous driving and mobile robot navigation, the uncertainty is rarely considered when deploying the trained policy. However, uncertainty is critical for robotics decision making. Unlike other pure perception scenarios, where higher uncertainty of the prediction may influence the accuracy of a segmentation mask or output an incorrect classification result, the uncertain decision in autonomous driving endangers the safety of vehicles or even human lives. Thus, we should not always assume that the output of the deep model is accurate. Knowing what a model does not understand is an essential part, especially for autonomous driving under dynamic environments and interacting with pedestrians and vehicles.

When arranging the policy in the testing world, like the real world, a common pipeline is translating the visual input from the real world back to the training simulation environment [2], [5] through a generative adversarial network (GAN). Most of the previous works have focused on image-to-image transfer through a deterministic generator. However, the imitation learning policy is usually trained in a multi-domain environment with various conditions for

This work was supported by the National Natural Science Foundation of China (Grant No. U1713211), the Research Grant Council of Hong Kong SAR Government, China, under Project No. 11210017, and No. 21202816, and Shenzhen Science, Technology and Innovation Comission (SZSTI) JCYJ20160428154842603, awarded to Prof. Ming Liu.

All authors are with The Hong Kong University of Science and Technology (email: {ltai, pyun, ychenco, cliubh, hyeab, eelium}@ust.hk).

better generalization. Thus, for a deterministic translation, the problem is which training scenario should we transfer the real-world image to. In this paper, we extend this pipeline to generate various translated images with training data styles through multimodal cross-domain mapping. To generate the transferred images, we can randomly sample style codes from a normal distribution or directionally encode the provided style images from the training domain. The content code is extracted from the real-world image collected from the mounted sensor in real-time. A decoder would take the content code and style codes as input to generate various stylized images.

Naturally, we can predict the actions and uncertainties of all the translated images through the proposed uncertainty-aware imitation learning network. Among the generated images, the most certain one will be considered to deploy to the agent.

We list the main contributions of our work as follows:

- We transfer the real driving image back to diverse images stylized under the familiar training environment through a stochastic generator so that the decision is made through multiple alternate options.
- The uncertainty-aware imitation learning network provides a considerable way to make driving decisions, which improves the safety of autonomous driving, especially in dynamic environments.
- We explain the aleatoric uncertainty from the view of the noisily labelled data samples.

II. Related Works

In this section, we review related works in end-to-end driving, uncertainty-aware decision making and visual domain adaptation.

A. End-to-end Driving

Traditional visual-based strategies in autonomous driving and robot navigation, traditional methods firstly recognise relevant objects from visual inputs, including pedestrians, traffic lights, lanes, and cars. That information is considered to make the final driving decisions based on manually designed rules [6]. Recently, benefits from the excellent approximation ability of deep neural networks, end-to-end methods have become more and more popular in vision-based navigation.

Tai *et al.* [4] used deep convolutional neural networks to map depth images to steering commands so that the agent can make meaningful decisions like a human demonstrator in an indoor corridor environment. A similar framework was also successfully applied in a forest trail scenario to navigate a flying platform for obstacle avoidance [7]. They also considered softly combining all the discrete commands based on the weighted outputs of the softmax structure. Codevilla *et al.* [3] designed a deep structure with multiple branches for end-to-end driving through imitation learning. Based on the high-level commands from the global path planner, outputs from the specific branch are applied to the mobile agent.

Reinforcement learning (RL) algorithms also show surprising effects in end-to-end navigation. Zhang *et al.* [8] explored the target-arriving ability of a mobile robot through RL based on a single depth image. Their policy can also quickly adapt to new situations through successor features. For autonomous driving, RL algorithms are also considered to train an intelligent agent through interaction with simulated environments like *Carla* [1]. Liang *et al.* [9] used the model weight trained through imitation learning as the initialization of their reinforcement learning policy, while Tai *et al.* [10] proposed to solve the socially compliant navigation problem through inverse RL. However, all of the methods above directly deploy the learned policy on related platforms. None of them considers the uncertainty of the decision.

B. Uncertainty in learning-based decision making

The uncertainty in deep learning is derived from the *Bayesian deep learning* [11] approaches, where *aleatoric* uncertainty and *epistemic* uncertainty are extracted through specific learning structures [12]. Recently, computer vision researchers have started to leverage those uncertainties on related applications, like balancing the weight of different loss items for multi-task visual perception [13]. The uncertainty estimation helps the deep *Bayesian* models to achieve state-of-the-art results on various computer vision benchmarks, including semantic segmentation and depth regression.

In terms of decision making in robotics, Kahn *et al.* [14] proposed an uncertainty-aware model-based reinforcement learning method to update the confidence for a specific obstacle iteratively. During the training phase, the agent behaves more carefully in unfamiliar scenarios at the beginning. Based on this work, Lutjens *et al.* [15] explored more complex pedestrian-rich environments. The uncertainty was further considered for the exploitation and exploration balance in their implementation [15]. Henaff *et al.* [16] focused on the out-of-distribution data where an uncertainty cost was used to represent the divergence of the test sample from the training states. However, all of the methods above follow a pipeline using multiple stochastic forward passes through *Dropout* to estimate the *epistemic* uncertainty [12]. The time-consuming computation potentially limits these methods in scenarios which ask for real-time deployment ability.

A highly related work is the work of Choi *et al.* [17]. They proposed a novel uncertainty estimation method where a single feedforward is enough for uncertainty acquisition. However, they only tested their method in the state space. In this paper, we aim to tackle a much more difficult visual-based navigation problem.

C. Visual domain adaptation

For a policy trained in simulated environments or based on datasets collected from simulated environments, the gap with the testing world (e.g. the real world) is always an essential problem. In the following, we mainly review the policy transferring methods through image translation.

One probable solution is the so-called *sim-to-real*, where synthetic images are translated to the realistic domain [18]. With an additional adaptation step for each training iteration, the whole training-deployment procedure is inevitably slowed down.

Another direction is *real-to-sim*, where real-world images are translated back to simulated environments. Zhang *et al.* [2] extended the *CycleGAN* [19] framework with a *shift loss*, which improves the consistency of the generated image streams. They achieved great improvements on the *Carla* [1] navigation benchmark. Muller *et al.* [20] firstly perceived a real-world RGB image as a segmentation mask which is used to generate path points through a learned policy network.

For the purely unsupervised image-to-image translation problem, unlike the previous deterministic translation model [19], multimodel mapping has received lots of attention from computer vision researchers [21], [22], [23]. Their goal is translating an image from the source domain to a conditional distribution of the related image in the target domain. This is a naturally applicable method for a robotic task because the training domain of the policy networks always contains data collected from various conditions (e.g. different weathers [3], [2]) for better generalization ability.

III. An explanation of *Aleatoric* uncertainty

As mentioned before, there are two types of uncertainty in deep learning as introduced in [11] and [12], the *aleatoric* uncertainty and the *epistemic* uncertainty. The *epistemic* uncertainty is the model uncertainty, which can be reduced by adding enough data. However, it is commonly realized through stochastic *Dropout* forward passes, which cost too much time to be applied in real time. In this paper, we mainly consider the *aleatoric* uncertainty, the data uncertainty.

Following the heteroscedastic *aleatoric* uncertainty setup in [12], a regression task can be represented as

$$[\tilde{\mathbf{y}}, \tilde{\sigma}] = \mathbf{f}^{\theta}(\mathbf{x}), \tag{1}$$

$$\mathcal{L}(\theta) = \frac{1}{2}\frac{\|\mathbf{y} - \tilde{\mathbf{y}}\|^2}{\tilde{\sigma}^2} + \frac{1}{2}\log\tilde{\sigma}^2. \tag{2}$$

Here, $\mathbf{x}$ is the input data, $\mathbf{y}$ and $\tilde{\mathbf{y}}$ are the groundtruth regression target and the predicted result. $\tilde{\sigma}$ is another output of the model and can represent the standard variance of the data $\mathbf{x}$, and θ is the model weight of the regression model.

We provide an explanation for $\tilde{\sigma}$ to show why it can represent the standard variance or the uncertainty of $\mathbf{x}$. Suppose that there is a subset Ψ of the training dataset, $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{N_\psi}$ with size N_ψ. For the prediction and the uncertainty, $\{\tilde{\mathbf{y}}_i, \tilde{\sigma}_i : [\tilde{\mathbf{y}}_i, \tilde{\sigma}_i] = \mathbf{f}^{\theta_\psi}(\mathbf{x}_i)\}_{i=1}^{N_\psi}$. The optimization target of this subset is

$$\min\mathcal{L}(\theta_\psi) = \min\sum_i^{N_\psi}\frac{1}{2}\frac{\|\mathbf{y}_i - \tilde{\mathbf{y}}_i\|^2}{\tilde{\sigma}_i^2} + \frac{1}{2}\log\tilde{\sigma}_i^2. \tag{3}$$

θ_ψ is the model weight to optimize for this subset Ψ. Assume that all the $\mathbf{x}_i$ in this subset are exactly the same, as $\mathbf{x}_\psi$. Because of the limitations of human labelling, they may be labelled with conflicting ground truths (like the noise labels around object boundaries in [12]). Then, the model will output the same prediction $\tilde{\mathbf{y}}_\psi$ and uncertainty $\tilde{\sigma}_\psi$ for all of $\{\mathbf{x}_i\}_{i=1}^{N_\psi}$ as $[\tilde{\mathbf{y}}_\psi, \tilde{\sigma}_\psi] = \mathbf{f}^{\theta_\psi}(\mathbf{x}_\psi)$. The minimization target turns to

$$\min\mathcal{L}(\theta_\psi) = \min\sum_i^{N_\psi}\frac{1}{2}\frac{\|\mathbf{y}_i - \tilde{\mathbf{y}}_\psi\|^2}{\tilde{\sigma}_\psi^2} + \frac{1}{2}\log\tilde{\sigma}_\psi^2. \tag{4}$$

Considering that $\tilde{\mathbf{y}}_\psi$ and $\tilde{\sigma}_\psi$ are conditionally independent on $\mathbf{x}_\psi$, $\tilde{\sigma}_\psi$ can be derived through the first-order derivative as

$$\begin{aligned}
\frac{\partial\mathcal{L}(\theta_\psi)}{\partial\tilde{\sigma}_\psi} &= \sum_i^{N_\psi} -\frac{\|\mathbf{y}_i - \tilde{\mathbf{y}}_\psi\|^2}{\tilde{\sigma}_\psi^3} + \frac{1}{\tilde{\sigma}_\psi} \\
&= \sum_i^{N_\psi} -\frac{\|\mathbf{y}_i - \tilde{\mathbf{y}}_\psi\|^2}{\tilde{\sigma}_\psi^3} + \frac{1}{\tilde{\sigma}_\psi} = 0 \\
\Rightarrow \sum_i^{N_\psi}\frac{\|\mathbf{y}_i - \tilde{\mathbf{y}}_\psi\|^2}{\tilde{\sigma}_\psi^3} &= \frac{N_\psi}{\tilde{\sigma}_\psi} \\
\frac{\sum_i^{N_\psi}\|\mathbf{y}_i - \tilde{\mathbf{y}}_\psi\|^2}{N_\psi} &= \tilde{\sigma}_\psi^2.
\end{aligned} \tag{5}$$

For the model $\mathbf{f}^{\theta_\psi}$, it makes sense to output $\mathbf{y}_\psi$ as the mean of $\{\mathbf{y}_i\}_{i=1}^{N_\psi}$. And that is why $\tilde{\sigma}_\psi^2$, as the prediction variance of the $\{\mathbf{y}_i\}_{i=1}^{N_\psi}$, can be regarded as the uncertainty of $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{N_\psi}$. For a decision-making task, even though at some point, the model cannot predict a good enough command, it should know this prediction is uncertain but not directly deploy it.

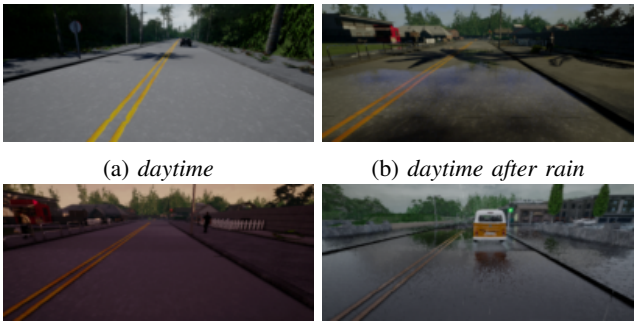
IV. Implementations

A. Carla navigation dataset and benchmark

As mentioned in [2], it is difficult to evaluate the autonomous driving policy under a common benchmark in the real world. Thus, we use the *Carla* driving dataset[1] to train the visual-based navigation policy. Then for the evaluation, we can naturally deploy it through the *Carla* navigation benchmark [1], [3] under an unseen extreme weather condition. The distribution of the *Carla* dataset [1] and the benchmark details are available in [24].

The collected expert dataset of *Carla* includes four different weather conditions (*daytime, daytime after rain, clear sunset and daytime hard rain*). The original experiments in [1] tested its policy under *cloudy daytime* and *soft rain at sunset*. However, considering these two weathers are not available in the provided dataset for domain adaptation, we resplit the *Carla* driving dataset into a training domain (*daytime, daytime after rain, clear sunset*) and testing domain (*daytime hard rain*), as shown in Fig. 2, following the setup in [2]. The vehicle speed, ground truth actions

[1]https://github.com/carla-simulator/imitation-learning



(a) *daytime* (b) *daytime after rain*

(c) *clear sunset* (d) *daytime hard rain*

Fig. 2: *Carla* weather conditions considered in this paper: training conditions including (a) *daytime*, (b) *daytime after rain* and (c) *clear sunset* and the testing condition (d) *daytime hard rain*.

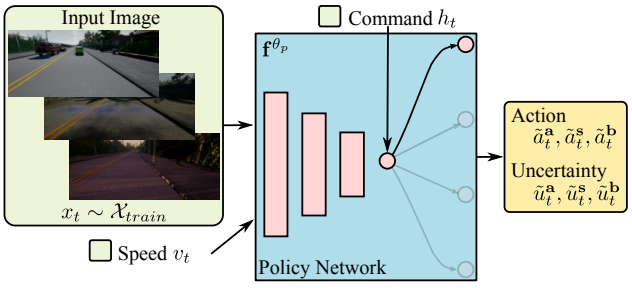


Fig. 3: The uncertainty-aware imitation learning pipeline of this paper. Based on the branched structure of the conditional imitation learning [3], a first-person-view image and the related velocity are taken as the input of the network. The final output is from the branch decided by the corresponding high-level command. The network also further generates uncertainties matching each output. The loss function is described in Section IV-B.

and related measurements are also provided by the dataset [1] and considered by our policy model.

The final testing environment under *daytime hard rain* is very challenging. We believe that the difficulty in deploying the policy through visual domain transformation from the testing domain to the training domain (*test-to-train*) in this paper can be regarded as comparable to the previous *real-to-sim* experiments [2], [5].

B. Uncertainty-aware Imitation Learning

We first introduce the framework of the policy network, which is the proposed uncertainty-aware imitation learning network, as shown in Fig. 3. The backbone of our policy network is based on the conditional imitation learning network [3] for visual-based navigation.

For this framework, the training dataset includes all three kinds of weather in the training domain mentioned in Section IV-A. In each forward step, an RGB Image x_t from the training dataset and the related vehicle's speed v_t are taken as the input to the network. The extracted features are passed to four different branches with the same structures. A high-level command h_t (*straight, left, right, follow line*) from the global path planner decides that output from which the branch will be chosen as the final prediction. The output consists of the predicted action $\tilde{a}_t$ and its estimated uncertainty $\tilde{\sigma}_t$. In practice, as in [12], we let the netowrk predict the log variance $\tilde{u}_t := \log \tilde{\sigma}_t^2$. The action $\tilde{a}_t$ is actually a vector including accelaration $\tilde{a}_t^{\mathbf{a}}$, steering $\tilde{a}_t^{\mathbf{s}}$, and braking $\tilde{a}_t^{\mathbf{b}}$, with their related uncertainties $\tilde{u}_t^{\mathbf{a}}$, $\tilde{u}_t^{\mathbf{s}}$, and $\tilde{u}_t^{\mathbf{b}}$ repectively.

We use θ_p to represent the weight of the policy network. $a_t^{\mathbf{a}}, a_t^{\mathbf{s}}$ and $a_t^{\mathbf{b}}$ represent the groundtruth actions from the collected dataset. The policy prediction process $\mathbf{f}^{\theta_p}$ and our uncertainty-aware loss function $\mathcal{L}_p$ are as follows:

$$[\tilde{a}_t^{\mathbf{a}}, \tilde{a}_t^{\mathbf{s}}, \tilde{a}_t^{\mathbf{b}}, \tilde{u}_t^{\mathbf{a}}, \tilde{u}_t^{\mathbf{s}}, \tilde{u}_t^{\mathbf{b}}] = \mathbf{f}^{\theta_p}(x_t, v_t, h_t) \tag{6}$$

$$\begin{aligned}\mathcal{L}_p(\theta_p) = &\frac{1}{2}\exp(-\tilde{u}_t^{\mathbf{a}})\|a_t^{\mathbf{a}} - \tilde{a}_t^{\mathbf{a}}\|^2 + \frac{1}{2}\tilde{u}_t^{\mathbf{a}} \\ &+ \frac{1}{2}\exp(-\tilde{u}_t^{\mathbf{s}})\|a_t^{\mathbf{s}} - \tilde{a}_t^{\mathbf{s}}\|^2 + \frac{1}{2}\tilde{u}_t^{\mathbf{s}} \\ &+ \frac{1}{2}\exp(-\tilde{u}_t^{\mathbf{b}})\|a_t^{\mathbf{b}} - \tilde{a}_t^{\mathbf{b}}\|^2 + \frac{1}{2}\tilde{u}_t^{\mathbf{b}}.\end{aligned} \tag{7}$$

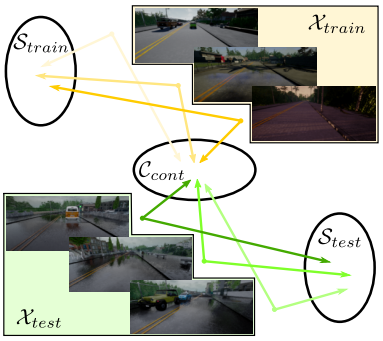


Fig. 4: The stochastic visual domain transformation structure following [21]. The training domain $\mathcal{X}_{train}$ consists of three weather conditions and the testing domain $\mathcal{X}_{test}$ is under a specific weather condition. These two domains share the same content space and maintain their own style space. The setting pipeline is explained in Section IV-C.

C. Stochastic real(test)-to-sim(train) Transformation

For the unsupervised *real-to-sim* pipeline, previous works [5], [2] are based on a deterministic structure like *CycleGAN* [19]. In this paper, we mainly consider stochastic multimodel translation [21] through GAN. The training domain, with three different kinds of weather, is represented as $\mathcal{X}_{train}$, and the testing domain, with a single weather condition, is represented as $\mathcal{X}_{test}$, as shown in Fig. 4. These two domains are supposed to maintain their distinguishable style space ($\mathcal{S}_{test}$ and $\mathcal{S}_{train}$) but share a common content space $\mathcal{C}_{cont}$. The stochastic model contains an encoder ($\mathbf{E}_{test}, \mathbf{E}_{train}$) and a decoder ($\mathbf{D}_{test}, \mathbf{D}_{train}$) for each of the domains. The training procedure follows the setup in [21]. For example, an image $x_1 \sim \mathcal{X}_{train}$ sampled from the training domain can be encoded to its style code s_1 and content code c_1 by the training domain encoder $\mathbf{E}_{train}$. The training domain decoder can also combine these two codes to generate $\tilde{x}_1$

as the reconstruction of x_1 as follows:

$$[c_1, s_1] = \mathbf{E}_{train}(x_1) \tag{8}$$

$$\tilde{x}_1 = \mathbf{D}_{train}(c_1, s_1). \tag{9}$$

For the cross-domain translation, the testing domain decoder $\mathbf{D}_{test}$ combines a random style code $\hat{s}_2$ from the testing domain and the content code c_1 to generate the translated image $\tilde{x}_{1\to2}$. The testing domain encoder $\mathbf{E}_{test}$ takes this translated image as input and generates $\tilde{c}_1$ and $\tilde{s}_2$ as the reconstruction of c_1 and $\hat{s}_2$ as follows:

$$\tilde{x}_{1\to2} = \mathbf{D}_{test}(c_1, \hat{s}_2) \tag{10}$$

$$[\tilde{c}_1, \tilde{s}_2] = \mathbf{E}_{test}(\tilde{x}_{1\to2}). \tag{11}$$

$(\hat{s}_2, \tilde{s}_2)$, $(c_1, \tilde{c}_1)$, and $(x_1, \tilde{x}_1)$ are constrained by L1 loss. A discriminator of the GAN structure is used to distinguish $\tilde{x}_{1\to2}$ from the original testing domain images. We skip the reconstruction of the testing domain image and the translation procedure from the testing domain to the training domain. The content code is a 2D matrix with a size corresponding to the input image. The style code is a vector with eight individually sampled numbers from the normal distribution. Note that the whole pipeline is unsupervised. Images from the two domains do not need to be paired for the training.

D. Deloyment phase

In the deployment phase, the final forward pipeline is shown in Fig. 1. We list all steps in Algorithm 1. After the training of all model weights, $\mathbf{f}^{\theta_p}$ (Section IV-B), $\mathbf{E}_{test}$, $\mathbf{E}_{train}$, and $\mathbf{D}_{train}$ (Section IV-C), the whole piepline can be deployed in the testing environment. In each time step, an image x_t collected from the sensor mounted on the vehicle in the *Carla* environment under the testing weather condition (*daytime hard rain*) is firstly taken to the encoder of the testing domain $\mathbf{E}_{test}$ to encode the content code c_t. The style codes $\{s'_{tj}\}_{j=1}^{M}$ can be encoded from the sampled training domain images $\{x'_{tj}\}_{j=1}^{M}$ by the training domain encoder $\mathbf{E}_{train}$, or directly sampled from a normal distribution. Through the training domain decoder $\mathbf{D}_{train}$, the original input image x_t is translated to various generated images $\{\tilde{x}_{tj}\}_{j=1}^{M}$ under different training domain styles. Those generated images will be processed by the pre-trained uncertainty-aware imitation learning policy network $\mathbf{f}^{\theta_p}$. Thus, we get actions and uncertainties corresponding to all the translated images. Among those actions, the one with the lowest uncertainty will be finally deployed to the mobile agent. Here, we skip the details of the actions $(\tilde{a}_t^{\mathbf{a}}, \tilde{a}_t^{\mathbf{s}}, \tilde{a}_t^{\mathbf{b}})$, which are all decided by their own uncertainties individually.

Algorithm 1 Real-time deployment pipeline

Pretained model weight $\mathbf{f}^{\theta_p}, \mathbf{E}_{test}, \mathbf{E}_{train}$ and $\mathbf{D}_{train}$.
// At testing time step t.
Get the real-time image $x_t \sim \mathcal{X}_{test}$.
Get the related velocity v_t and high-level command h_t.
Encode the content code of x_t: $[c_t, -\,] = \mathbf{E}_{test}(x_t)$.
if Encode style codes from images in $\mathcal{X}_{train}$, **then**
  Sample off-line images $\{x'_{tj}\}_{j=1}^{M} \sim \mathcal{X}_{train}$.
  Encode style codes $\{s'_{tj}\}_{j=1}^{M}$ of the selected style images: $[-, s'_{tj}] = \mathbf{E}_{train}(x'_{tj})$.
else
  Randomly sample style codes $\{s'_{tj}\}_{j=1}^{M}$ from the normal distribution.
end if
Images translation $\{\tilde{x}_{tj}\}_{j=1}^{M} : \tilde{x}_{tj} = \mathbf{D}_{train}(c_t, s'_{tj})$.
Generate actions through the policy network $\{\tilde{a}_{tj}, \tilde{u}_{tj}\}_{j=1}^{M} : [\tilde{a}_{tj}, \tilde{u}_{tj}] = \mathbf{f}^{\theta_p}(\tilde{x}_{tj}, v_t, h_t)$.
Locate the minimal uncertainty: $j^* = \mathrm{argmin}_j\{\tilde{u}_{tj}\}_{j=1}^{M}$.
Output and deploy $\tilde{a}_{tj^*}$.

V. Experiments

A. Model Training

For the stochastic translation model training, we follow the setup in [21][2] for $1e6$ steps with batch size as 1. As we mentioned before, the training domain $\mathcal{X}_{train}$ consists of three weather conditions and the testing domain $\mathcal{X}_{test}$ only contains the images under *daytime hard rain*. The size of original images in the *Carla* dataset is 200×88. To maintain enough information in the content code, they are resized to 256×256 for the stochastic image translation. After that, we get the trained encoders $(\mathbf{E}_{test}, \mathbf{E}_{train})$ and decoders $(\mathbf{D}_{test}, \mathbf{D}_{train})$, and they are used in the final forward pipeline, as shown in Section IV-D.

The training of uncertainty-aware imitation learning follows the branched structure of conditional imitation learning [3], [24]. We train all the training domain images (481600 images under three kinds of weather) for 90 epochs with a batch size of 1000. As in the original setup in [3], we also try several different network structures for the uncertainty estimation. Experiments show that the current structure processing the feature of the image and the velocity through another four branches outputting uncertainties of actions corresponding to the four high-level commands is the most effective. The code for the imitation learning policy training is available online.[3] We implement all the code through *Pytorch* and all the training is finished by an NVIDIA 1080Ti GPU.

B. Model Evaluation

Finally, we conduct experiments on the *Carla* navigation benchmark mentioned in Section IV-A. We compare different strategies both for the policy model and the visual domain transformation methods as follows:

For the policy model, we compare two different setups:

- **CIL**: Map the state to the action without considering the uncertainty as the original conditional imitation learning structure [3]. The output is directly deployed to the vehicle.
- **UAIL**: Take the uncertainty as an output of the network as described in Section IV-B. When using

[2]https://github.com/NVlabs/MUNIT

[3]https://github.com/onlytailei/carla_cil_pytorch

Policy model		CIL		UAIL				
Visual trans. model		Direct	CycleGAN	Direct	CycleGAN	Stoc.-Single.	Stoc.-Random.	Stoc.-Cross
Sucess rate(%)		0.0/-	34.7/-	14.7/16.0	44.0/56.0	50.1/60.0	54.7/**64.0**	**60.0/64.0**
Ave. distance to goal travelled(%)		5.2/-	55.7/-	25.8/29.5	66.4/78.0	73.0/76.3	62.1/67.8	**75.8/79.3**
Ave. distance	Opposite lane	0.26/-	0.83/-	4.43/6.23	1.44/1.77	6.91/11.05	11.66/20.67	**12.74/24.58**
travelled between	Sidewalk	0.38/-	1.29/-	1.26/1.68	2.10/3.16	3.72/4.42	3.79/5.17	**5.78/7.70**
two infractions	Collision-static	0.16/-	0.77/-	0.37/0.52	1.22/1.75	2.46/3.16	3.03/6.20	**9.56/23.09**
in Nav. dynamic	Collision-car	0.27/-	0.59/-	0.56/0.67	0.75/0.99	1.78/**2.15**	0.61/0.64	**2.04**/2.14
(km)	Collision-ped.	4.60/-	7.29/-	0.39/0.61	8.07/8.85	8.17/11.04	5.95/10.34	**9.87/23.58**

TABLE I: Quantitative experiments of the *Carla* dynamic navigation benchmark [1]. For the imitation learning policy, we compare our *UAIL* with the multi-domain (*CIL*) policy results in [2]. The proposed uncertainty-aware imitation learning policy overtakes the original *CIL* in all the metrics under both direct deployment (*Direct*) and deterministic transformation through *CycleGAN*. For different visual domain adaptation methods under the *UAIL* policy model, the stochastic methods show much better performances compared with *Direct* and *CycleGAN*. Among them, the proposed *Stocastic-Cross*, considering both the randomization and the directional training styles, achieves the best performance. All the results are shown as the average/max values of the three benchmark trials. Higher means better for all metrics.

multiple visual inputs, the action with the lowest uncertainty is chosen as the final command.

For the visual domain adaptation methods, five strategies are compared:

- **Direct**: Directly deploy the control policy in the testing environment without any visual domain adaptations.
- **CycleGAN**: Transfer the real-time image to a specific training condition through *CycleGAN* [19] deterministically.
- **Stochastic-Single**: Directionally transfer the input image to a specific training weather condition based on the style image from the training domain.
- **Stochastic-Random**: Randomly sample three style codes to decode the translated images.
- **Stochastic-Cross**: Directionally transfer the real-time image to all the three training weather conditions based on the style images from the training domain.

For the *Stochastic-Single* and *Stochastic-Cross* transfer methods, the style codes are encoded from style images in the training domain. We prepare ten images for each of the training weather conditions. They are randomly sampled from the training dataset under the related weather condition. In each step, *Stochastic-Single* samples one style image from the related weather condition and *Stochastic-Cross* samples three style images from each of the training conditions, respectively.

The *Carla* navigation benchmark consists of four tasks, *Straight*, *One turn*, *Navigation* and *Navigation with dynamic obstacles*, where the vehicle need to finish 25 different navigation routes in each task. Since the first three tasks do not consider any pedestrians or vehicles in the environment, previous methods [3], [2] have achieved considerable generalization results on these tasks. However, in this paper, we mainly consider the most challenging task, *Navigation with dynamic obstacles*, under the testing weather condition.[4]

We run each of the setups three times and show the average/max result through the related benchmarks in Table I. As a deterministic transfer method, we build three transfer models between each training weather condition and the testing weather condition through *CycleGAN*. In each benchmark trial, one specific transfer model is used. The stochastic model can generate various stylized images through a batch operation. However, to achieve such processing through *CycleGAN*, we need to input the real-time image to each of the deterministic transfer models one-by-one, which is both time and resource-consuming. So the three benchmark experiments on *CycleGAN* are under a specific training weather condition for each time. It is the same for the *Stochastic-Single* method, except that the specific training style image is sampled from the prepared subset with a size of ten, as mentioned before.

We do not show the result of combining the *CIL* policy model with the stochastic transfer models because, without uncertainties, there is no reason to choose the specific one among the actions generated through various input images. The results of *CIL* policy are referred from [2], where their *multi-domain* policy is what we mean by *CIL* here. The results in [2] do not provide the max value of their trials.

Among the different policy models under *Direct* deployment and transformation with *CycleGAN*, our proposed *UAIL* shows great improvements in all of the metrics of the *Carla* navigation benchmark. In the comparison of different transformation methods, *Stochastic-Cross* shows the best generalization under the testing weather condition.

To understand the selection mechanism under our proposed *UAIL* and *Stochastic-Cross* pipeline, we show two typical uncertainty estimation examples during the testing in Fig. 5. As shown in Fig. 5-I, the outputs of actions and uncertainties between different translated images are quite close to each other, even though we choose the actions from the last image based on the lowest uncertainty. Since the straight line scenario is the most common one in the training dataset and the decision is relatively simple to make. The dynamic and challenging turning scenario in Fig. 5-II is what we are particularly aiming to solve. The second transferred image under *Clear Sunset* outputs a very tiny steering command which could potentially cause a collision with the car in front.

[4]The uncertainty-aware policy is a little bit conservative, so we relax the time limit for each trial. However, this does not affect the results of infractions in Table. I

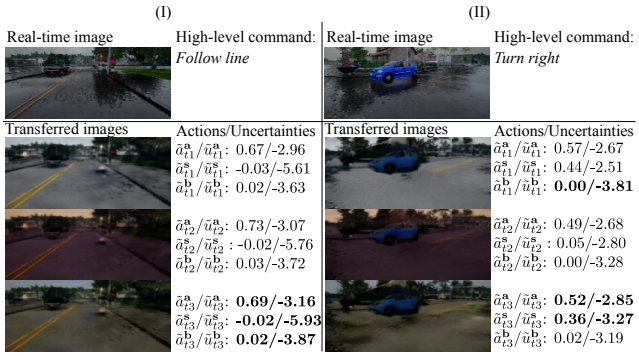


Fig. 5: Two examples for the proposed pipeline *UAIL* with *Stochastic-Cross*. The final commands with the lowest uncertainties are labelled. For the straight line in (I), which is a relatively clean environment, the uncertainties from all the generated images are almost the same. For the more complex dynamic environment with the turning condition in (II), the chosen steering command $\tilde{a}^{\mathbf{s}}_{t3}$ is much safer. Without an effective steering command, a collision will happen like the $\tilde{a}^{\mathbf{s}}_{t2}$.

VI. Conclusions

We proposed a deployment pipeline for a visual-based navigation policy under the *real-to-sim* structure. Through considering the *aleatoric* data uncertainty and the stochastic transformation when translating the testing image back to the training domain, a safer action selection mechanism is constructed for end-to-end driving. Experiments on deploying the pre-trained policy in an unknown extreme weather condition through the *Carla* navigation benchmark show that our proposed pipeline provides a more certain and robust solution.

For future work, finally transferring the trained policy to real-world autonomous driving in a challenging environment would be an exciting next step. Considering the consistency of image streams, like in [2], could be another future direction. Furthermore, this alternative decision-making pipeline may also guide the improvement of the model training for challenging samples. Related model augmentation towards more robust generalization could also be performed.

References

[1] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *CoRL*, vol. 78. PMLR, 13–15 Nov 2017, pp. 1–16.

[2] J. Zhang, L. Tai, P. Yun, Y. Xiong, M. Liu, J. Boedecker, and W. Burgard, "Vr-goggles for robots: Real-to-sim domain adaptation for visual control," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1148–1155, April 2019.

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

[4] L. Tai, S. Li, and M. Liu, "A deep-network solution towards model-less obstacle avoidance," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 2759–2764.

[5] L. Yang, X. Liang, T. Wang, and E. Xing, "Real-to-virtual domain unification for end-to-end autonomous driving," in *ECCV*. Cham: Springer International Publishing, 2018, pp. 553–570.

[6] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[7] A. Giusti, J. Guzzi, D. C. Cirean, F. He, J. P. Rodrguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. D. Caro, D. Scaramuzza, and L. M. Gambardella, "A machine learning approach to visual perception of forest trails for mobile robots," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 661–667, July 2016.

[8] J. Zhang, J. T. Springenberg, J. Boedecker, and W. Burgard, "Deep reinforcement learning with successor features for navigation across similar environments," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep 2017, pp. 2371–2378.

[9] X. Liang, T. Wang, L. Yang, and E. Xing, "Cirl: Controllable imitative reinforcement learning for vision-based self-driving," in *The European Conference on Computer Vision (ECCV)*, September 2018.

[10] L. Tai, J. Zhang, M. Liu, and W. Burgard, "Socially compliant navigation through raw depth inputs with generative adversarial imitation learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 1111–1117.

[11] Y. Gal, "Uncertainty in deep learning," Ph.D. dissertation, University of Cambridge, 2016.

[12] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" in *Advances in neural information processing systems*, 2017, pp. 5574–5584.

[13] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7482–7491.

[14] G. Kahn, A. Villaflor, V. Pong, P. Abbeel, and S. Levine, "Uncertainty-aware reinforcement learning for collision avoidance," *arXiv preprint arXiv:1702.01182*, 2017.

[15] B. Lütjens, M. Everett, and J. P. How, "Safe reinforcement learning with model uncertainty estimates," *arXiv preprint arXiv:1810.08700*, 2018.

[16] M. Henaff, A. Canziani, and Y. LeCun, "Model-predictive policy learning with uncertainty regularization for driving in dense traffic," *arXiv preprint arXiv:1901.02705*, 2019.

[17] S. Choi, K. Lee, S. Lim, and S. Oh, "Uncertainty-aware learning from demonstration using mixture density networks with sampling-free variance modeling," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 6915–6922.

[18] X. Pan, Y. You, Z. Wang, and C. Lu, "Virtual to real reinforcement learning for autonomous driving," in *Proceedings of the British Machine Vision Conference (BMVC)*, 2017.

[19] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2223–2232.

[20] M. Mueller, A. Dosovitskiy, B. Ghanem, and V. Koltun, "Driving policy transfer via modularity and abstraction," in *Proceedings of The 2nd Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, A. Billard, A. Dragan, J. Peters, and J. Morimoto, Eds., vol. 87. PMLR, 29–31 Oct 2018, pp. 1–15. [Online]. Available: http://proceedings.mlr.press/v87/mueller18a.html

[21] X. Huang, M.-Y. Liu, S. Belongie, and J. Kautz, "Multimodal unsupervised image-to-image translation," in *The European Conference on Computer Vision (ECCV)*, September 2018.

[22] H.-Y. Lee, H.-Y. Tseng, J.-B. Huang, M. Singh, and M.-H. Yang, "Diverse image-to-image translation via disentangled representations," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 35–51.

[23] A. Almahairi, S. Rajeswar, A. Sordoni, P. Bachman, and A. Courville, "Augmented cyclegan: Learning many-to-many mappings from unpaired data," *arXiv preprint arXiv:1802.10151*, 2018.

[24] J. Zhang, L. Tai, Y. Xiong, M. Liu, J. Boedecker, and W. Burgard, "Supplement file of VR-Goggles for robots: Real-to-sim domain adaptation for visual control," Tech. Rep., 2018. [Online]. Available: https://ram-lab.com/file/tailei/vr_goggles/supplement.pdf