IEEE Robotics and Automation Letters (RAL) paper presented at
2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Seamless Virtual Reality With Integrated Synchronizer and Synthesizer for Autonomous Driving

He Li, *Graduate Student Member, IEEE*, Ruihua Han, *Graduate Student Member, IEEE*, Zirui Zhao, Wei Xu, Qi Hao, *Member, IEEE*, Shuai Wang, *Senior Member, IEEE*, and Chengzhong Xu, *Fellow, IEEE*

***Abstract*—Virtual reality (VR) is a promising data engine for autonomous driving (AD). However, data fidelity in this paradigm is often degraded by VR inconsistency, for which the existing VR approaches become ineffective, as they ignore the inter-dependency between low-level VR synchronizer designs (i.e., data collector) and high-level VR synthesizer designs (i.e., data processor). This paper presents a seamless virtual reality (SVR) platform for AD, which mitigates such inconsistency, enabling VR agents to interact with each other in a shared symbiotic world. The crux to SVR is an integrated synchronizer and synthesizer (IS^2) design, which consists of a drift-aware lidar-inertial synchronizer for VR colocation and a motion-aware deep visual synthesis network for augmented reality image generation. We implement SVR on car-like robots in two sandbox platforms, achieving a cm-level VR colocalization accuracy and 3.2% VR image deviation, thereby avoiding missed collisions or model clippings. Experiments show that the proposed SVR reduces the intervention times, missed turns, and failure rates compared to other benchmarks. The SVR-trained neural network can handle unseen situations in real-world environments, by leveraging its knowledge learnt from the VR space.**

***Index Terms*—Autonomous driving, imitation learning, sim-to-real, virtual reality.**

Manuscript received 8 November 2023; accepted 26 February 2024. Date of publication 14 March 2024; date of current version 25 March 2024. This letter was recommended for publication by Associate Editor A. Thakur and Editor A. Banerjee upon evaluation of the reviewers' comments. This work was supported in part by the Science and Technology Development Fund of Macao S.A.R (FDCT) under Grant 0123/2022/AFJ and Grant 0081/2022/A2, in part by the National Natural Science Foundation of China under Grant 62371444 and Grant 62261160654, and in part by Shenzhen Fundamental Research Program under Grant JCYJ20220818103006012. *(Corresponding authors: Shuai Wang; Chengzhong Xu.)*

He Li and Chengzhong Xu are with the State Key Laboratory of Internet of Things for Smart City (SKL-IOTSC), University of Macau, Macau 999078, China (e-mail: mc25094@um.edu.mo; czxu@um.edu.mo).

Ruihua Han is with the Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518000, China, and also with the Department of Computer Science, The University of Hong Kong, Hong Kong (e-mail: hanrh@connect.hku.hk).

Zirui Zhao and Qi Hao are with the Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518000, China (e-mail: zhaozr@mail.sustech.edu.cn; hao.q@sustech.edu.cn).

Wei Xu is with the Manifold Tech Limited, Hong Kong SAR, China (e-mail: xuwei@manifoldtech.cn).

Shuai Wang is with the Center for Cloud Computing, Shenzhen Institute of Advanced Technology (SIAT), Chinese Academy of Sciences, Shenzhen 518000, China (e-mail: s.wang@siat.ac.cn).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2024.3375266, provided by the authors.

Digital Object Identifier 10.1109/LRA.2024.3375266

I. Introduction

Scarcity of boundary data (e.g., cut-in, crash) in real-world environments has led to incompleteness of the datasets used for design and validation of autonomous driving (AD) systems [1], [2], [3]. This makes it a challenge to push AD from closed to open scenarios. Compared to road tests, virtual simulation [2], [3], [4], [5], [6], which is based on modern computer graphics and physical modeling technologies, is a safer, faster, and cheaper method that can generate boundary scenarios that are difficult to be produced in physical environments. However, there exists a non-negligible gap between the simulated and real-world data [1], [5], [7].

To mitigate the gap, a promising solution is to leverage virtual reality (VR) (i.e., defined as a pair of physical and digital worlds in this paper) for more realistic data generation. Conventional VRAD schemes [7], [8], [9] process off-the-shelf datasets using some sim-to-real transfer methods, which fail in collecting end-to-end data (e.g., no action-in-the-loop). Emerging interactive VRAD [10], [11], [12], [13] leverages vehicle-in-the-loop (VIL) or environment-in-the-loop (EIL) to collect end-to-end augmented reality (AR) data, while ensuring exact vehicle dynamics and road environments. Nonetheless, VIL or EIL approaches may break down whenever VR inconsistency emerges, due to low-frequency colocation (e.g. RTK-GPS colocation [10], [11]), incomplete interaction (e.g., no EIL [12], [13]), or lack of reciprocal considerations between low-level VR designs (i.e., data collector) and high-level VR designs (i.e., data processor) [8], [9], [10], [11], [12], [13]. Such inconsistency would result in degraded data fidelity, e.g., missed collisions or model clippings.

To fill in the blank, this paper presents a seamless virtual reality (SVR) platform with integrated synchronizer and synthesizer (IS^2) to maximize the VR data fidelity (as shown in Fig. 1). We first propose a drift-aware lidar-inertial synchronizer (LIS) based on regularized error-state Kalman filter for VR colocation. With LIS, a pair of aligned VR poses (i.e., left hand side of Fig. 1) and the associated control vectors (i.e., right hand side of Fig. 1) are simultaneously collected at high frequency. To obtain aligned VR video streams and augmented images, this paper further proposes a motion-aware deep visual synthesis network (DVSN), which leverages neighbourhood consensus network (NC-Net) for image registration and segment anything model (SAM) for image aggregation. The DVSN effectively separates front objects from background in the virtual images and automatically calibrates objects' positions and sizes so as to best merge them into the real images for AR image generation

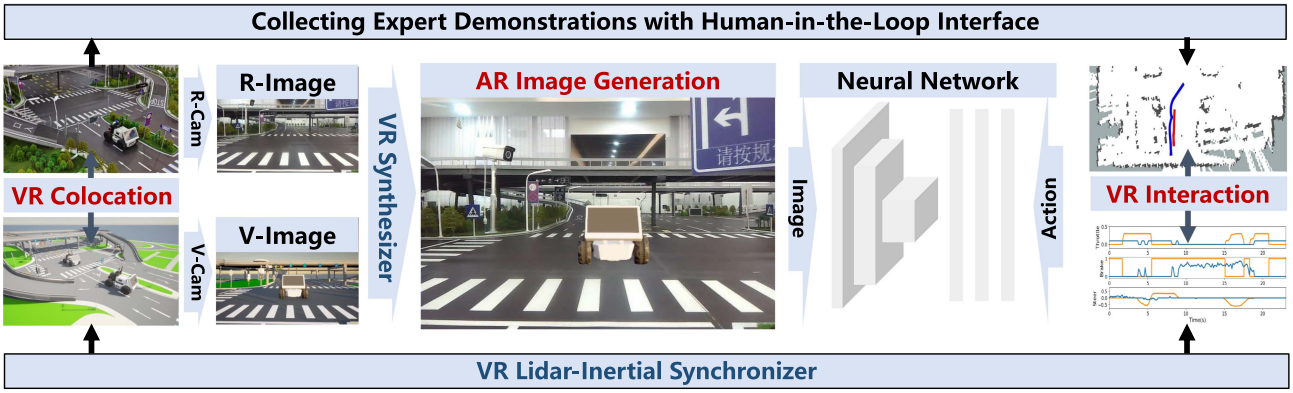


Fig. 1. Illustration of SVR with IS^2. A vehicle navigates in the physical space while interacting with obstacle vehicles in the digital space. The SVR generated dataset is used to train AD neural networks.

(i.e., middle of Fig. 1). Various experiments in robot operation system (ROS) show that LIS and DVSN achieve cm-level colocalization errors and minimum image deviations, enabling vehicles to simultaneously interact with physical and digital objects smoothly in a symbiotic SVR world. Experiments on a passenger vehicle in a parking lot are also provided to validate the effectiveness of SVR in real-world settings.

Furthermore, while VR data can augment the AD dataset, limited experimental evaluations have been reported for VR-trained imitation learning (IL) (a widely used AD paradigm). To this end, a human-in-the-loop interface (i.e., upper of Fig. 1) is used to collect expert driving datasets, and a conditional IL network is trained and tested (i.e., middle of Fig. 1) on car-like robots in two sandbox platforms. Experiments show that the proposed SVR trained IL reduces the intervention times, missed turns, and failure rates compared to benchmarks trained by real data, virtual simulation, or conventional VR methods. Interestingly, it is found that the SVR-trained IL can handle unseen situations in real environments, by leveraging its knowledge learnt from the VR space.

Our contributions are summarized as follows:

- Propose a drift-aware LIS, which accurately maps autonomous vehicles in the real world and adversarial obstacles in the virtual world to a symbiotic world;
- Propose a motion-aware DVSN, which automatically calibrates and segments objects from virtual images for seamless synthesis with environments from real images;
- Implementation of SVR-trained IL and evaluation of the performance gain brought by SVR to IL.

II. Related Work

Virtual simulation can generate corner cases that are difficult to be collected in the real world [4]. This technique adopts unreal engine (UE) for photo-realistic rendering (e.g., ray-casting) and physical engine (PE) for dynamics modeling (e.g., crash). Various AD simulators have been released, e.g., Intel Carla, ROS Gazebo, NVIDIA Drive, Waymo CarCraft, Tesla Autopilot, LG SVL, Tecent TAD-SIM, THU DAIR-V2X, UCLA OpenCDA, SJTU V2X-SIM, SIAT CarlaFLCAV [4], [5], [6]. Nevertheless, the gap between real and virtual worlds is non-negligible, which significantly impedes the credibility of AD simulation.

VRAD, which integrates VR and AD techniques, have been adopted to mitigate the above gap. VRAD can be categorized into non-interactive [7], [8], [9] and interactive approaches [10], [11], [12], [13]. Non-interactive VRAD approaches [7], [8], [9] focus on processing off-the-shelf datasets instead of generating interactive datasets, which are suitable for perception data augmentation instead of end-to-end data augmentation. For instance, generative adversarial networks (GAN) [8] have been adopted for domain adaption. Waymo and Google developed SurfelGAN, which can generate VR driving videos from real historical clips [9]. Baidu proposed augmented autonomous driving simulation (AADS) system, which generates augmented sensor data under different illuminations and weathers from ApolloScape [7].

Interactive VRAD, on the other hand, leads to a paradigm shift where previous isolated virtual and real agents can now naturally cooperate and compete in a shared driving scenario [14]. While interactive VR has been studied in various human-robot interaction tasks [14], [15], researches on interactive VRAD are still in its infancy. Recently, Intel, UCLA, and UMICH have released interactive VRAD platforms. For instance, Intel released Carla-ROS-Bridge interfaces for VIL simulation [13], and UCLA further bridged Carla-ROS with OpenCDA [12]. UMICH developed a VRAD platform that allows real vehicles in the physical test track to interact with virtual obstacles [11], and adopted this platform to train a reinforcement learning network for AD [10].

Nonetheless, the above interactive VRAD systems involve VR inconsistency as mentioned in Section I. Here, we present an IS^2 approach that achieves SVR by joint design of low-level VR (i.e., data collector) and high-level VR (i.e., data processor). We also demonstrate the effectiveness of SVR for IL with extensive real-world experiments, and for the first time, the performance gain brought by VRAD to IL is concisely quantified.

III. The SVR System

The architecture of SVR is shown in Fig. 2, which adopts the LIS for VR colocation, the DVSN for AR image generation, and the ROS interface for collecting end-to-end interactive datasets to train the IL network.

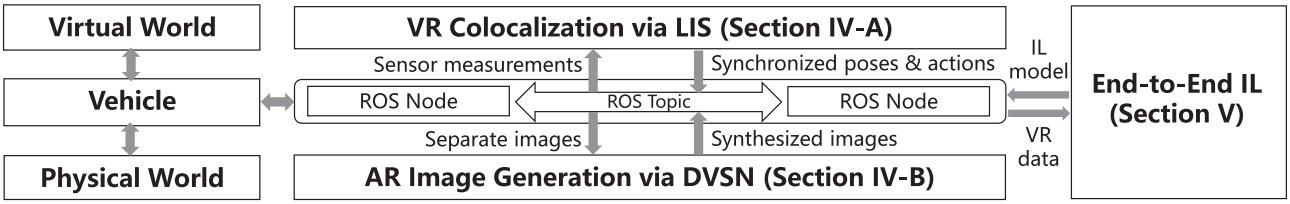


Fig. 2. System architecture of SVR, which integrates LIS, DVSN, and IL, through ROS.

The LIS module simultaneously estimates the vehicle poses ${}^{V}\mathbf{s}_t = ({}^{V}x_t, {}^{V}y_t, {}^{V}\theta_t)$ and ${}^{R}\mathbf{s}_t = ({}^{R}x_t, {}^{R}y_t, {}^{R}\theta_t)$ in the virtual and real spaces, respectively, where t is the index of lidar time frame, (x_t, y_t) and θ_t are positions and orientations, of the virtual vehicle (if the upper-left subscript is V) and the real vehicle (if the upper-left subscript is R). The two poses are connected through a rigid transformation with the extrinsic parameters of rotation matrix ${}^{VR}\mathbf{R}_t$ and translation vector ${}^{VR}\mathbf{t}_t$ between the two worlds. The initial guesses of extrinsics are ${}^{VR}\mathbf{R}_t^{[0]}$, ${}^{VR}\mathbf{t}_t^{[0]}$, which can be obtained using offline calibration. The VR colocation needs to update state vectors ${}^{V}\mathbf{s}_t$, ${}^{R}\mathbf{s}_t$ and their extrinsics ${}^{VR}\mathbf{R}_t$, ${}^{VR}\mathbf{t}_t$ as the vehicle moves in the VR space. In our system, this is realized based on lidar measurements ${}^{V}\mathbf{z}_t$, ${}^{R}\mathbf{z}_t$, and inertial measurements ${}^{V}\mathbf{w}_t$, ${}^{R}\mathbf{w}_t$ at the virtual and real vehicles. As such, the colocation algorithm can operate at 50 Hz even at low-cost vehicles in GPS-denied environments.

With colocation results from LIS, a sequence of paired images ${}^{V}\mathcal{U} = \{{}^{V}\mathbf{u}_1, {}^{V}\mathbf{u}_2, \cdots\}$ and ${}^{R}\mathcal{U} = \{{}^{R}\mathbf{u}_1, {}^{R}\mathbf{u}_2, \cdots\}$, as well as corresponding real-world actions $\mathcal{A} = \{\mathbf{a}_1, \mathbf{a}_2, \cdots\}$, are obtained, where $\mathbf{u}_k \in \mathbb{R}^{LW\times 1}$ denotes the real image (if the upper-left subscript is R) and its associated virtual image (if the upper-left subscript is V) of the k-th image frame with L and W being the image length and width, and $\mathbf{a}_k$ denotes the action vector containing steer, throttle, and brake elements. The DVSN module defines a mapping that fuses virtual image ${}^{V}\mathbf{u}_k$ and real image ${}^{R}\mathbf{u}_k$ into augmented image ${}^{A}\mathbf{u}_k = f_{\mathsf{DVSN}}({}^{V}\mathbf{u}_k, {}^{R}\mathbf{u}_k)$. The resultant end-to-end interactive dataset is given by ${}^{A}\mathcal{U} = \{{}^{A}\mathbf{u}_1, {}^{A}\mathbf{u}_2, \cdots\}$ and $\mathcal{A} = \{\mathbf{a}_1, \mathbf{a}_2, \cdots\}$.

The IL module is an inference mapping $f_{\mathsf{IL}} : \{{}^{R}\mathcal{U}, \mathcal{C}\} \to \mathcal{A}$ that maps real images and branch indicators into actions, where $\mathcal{C} = \{c_1, c_2, \cdots\}$, with c_k being a high-level command and acting as a switch that selects a certain back-end network At the k-th image frame, we have $\mathbf{a}_k = f_{\mathsf{IL}}({}^{R}\mathbf{u}_k, c_k|\mathbf{y}^*)$, where $\mathbf{y}^*$ is the IL weights trained with the expert demonstrations. In conventional AD systems, the expert demonstrations are collected from the real-world environment and $\mathbf{y}$ is trained on the real-world dataset $\{{}^{R}\mathcal{U}, \mathcal{A}\}$. In our system, however, the expert demonstrations are collected in the VR space and $\mathbf{y}^*$ is obtained from the VR dataset $\{{}^{A}\mathcal{U}, \mathcal{A}\}$.

IV. The IS^2 Approach

A. *VR Colocation With Drift-Aware* LIS

The LIS uses IMU measurements and features extracted in two consecutive virtual-reality scans to estimate the relative transformation of the vehicle and the extrinsic parameters between the VR space. Let ${}^{R}\mathbf{x}_t$ denote the vehicle state transformation from lidar time-step t to $t+1$ in the real world:

$$ {}^{R}\mathbf{x}_{t,t+1} = \left[{}^{R}\mathbf{p}_{t,t+1}, {}^{R}\mathbf{v}_{t,t+1}{}^{R}\mathbf{q}_{t,t+1}, {}^{R}\mathbf{b}_a, {}^{R}\mathbf{b}_g, {}^{R}\mathbf{g}_t\right], \quad (1) $$

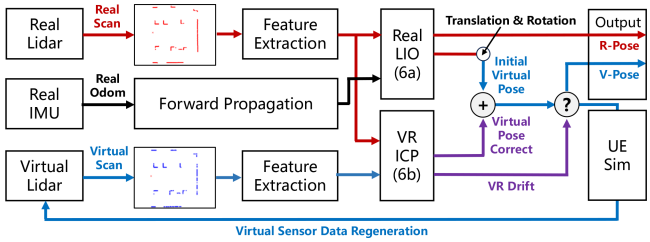


Fig. 3. Structure of LIS for VR colocation.

where ${}^{R}\mathbf{p}_{t,t+1}$ and ${}^{R}\mathbf{q}_{t,t+1}$ represent the translation and quaternion from time t to $t+1$, ${}^{R}\mathbf{v}_{t,t+1}$ is the velocity, ${}^{R}\mathbf{b}_a$ is the acceleration bias, ${}^{R}\mathbf{b}_g$ is the gyroscope bias, and ${}^{R}\mathbf{g}_t$ is the local gravity. All the above states are defined w.r.t. the local frame (i.e., IMU-affixed frame). We adopt error-state representation and the error vector of ${}^{R}\mathbf{x}_{t,t+1}$ is given by

$$ \delta\mathbf{x} = [\delta\mathbf{p}, \delta\mathbf{v}, \delta\theta, \delta\mathbf{b}_a, \delta\mathbf{b}_g, \delta\mathbf{g}], \quad (2) $$

where $\delta\theta$ is the error of angle ${}^{R}\theta_{t,t+1}$ associated with ${}^{R}\mathbf{q}_{t,t+1}$. With (2), we compute the state posterior ${}^{R}\mathbf{x}^{\text{post}}_{t,t+1}$ by injecting the error term $\delta\mathbf{x}$ into the state prior ${}^{R}\mathbf{x}^{\text{pri}}_{t,t+1}$ as:

$$ \begin{aligned} {}^{R}\mathbf{x}^{\text{post}}_{t,t+1} &= {}^{R}\mathbf{x}^{\text{pri}}_{t,t+1} \boxplus \delta\mathbf{x} \\ &= \Big[{}^{R}\mathbf{p}^{\text{pri}}_{t,t+1} + \delta\mathbf{p},\ {}^{R}\mathbf{v}^{\text{pri}}_{t,t+1} + \delta\mathbf{v},\ {}^{R}\mathbf{q}^{\text{pri}}_{t,t+1} \otimes \exp(\delta\theta), \\ &\quad {}^{R}\mathbf{b}^{\text{pri}}_{a} + \delta\mathbf{b}_a,\ {}^{R}\mathbf{b}^{\text{pri}}_{g} + \delta\mathbf{b}_g,\ {}^{R}\mathbf{g}^{\text{pri}}_{t} + \delta\mathbf{g}\Big], \end{aligned} \quad (3) $$

where $\boxplus$ is defined in the second line of (3), $\otimes$ is the quaternion product, and exp maps the angle to quaternion.

Based on the above analysis, the key of LIS is to estimate the error term $\delta\mathbf{x}$. However, in contrast to existing state estimation methods [16], [17] that only match the error term with the real-world sensor measurements, the VR colocation needs to further incorporate the virtual measurements into the state estimation for VR drift computations. This is because the goal of VR colocation is to collect synchronized VR measurements instead of accurate localization. To this end, we present a VR iterated Kalman filter approach shown in Fig. 3, where its forward propagation that outputs state prior is a standard approach (detailed in [16], [17]), but its backward propagation that outputs state posterior is different from existing approaches.

Specifically, since the virtual vehicle is a digital twin of the real vehicle, we have ${}^{V}\mathbf{x}_{t,t+1} = H({}^{R}\mathbf{x}_{t,t+1})$, where H is

determined by ${}^{VR}\mathbf{R}_t$ and ${}^{VR}\mathbf{t}_t$ as follows:

$$\begin{cases} {}^{V}\mathbf{p}_{t,t+1} &= {}^{VR}\mathbf{R}_t\, {}^{R}\mathbf{p}_{t,t+1} + {}^{VR}\mathbf{t}_t, \\ {}^{V}\mathbf{v}_{t,t+1} &= {}^{VR}\mathbf{R}_t\, {}^{R}\mathbf{v}_{t,t+1} + {}^{VR}\mathbf{t}_t, \\ {}^{V}\mathbf{q}_{t,t+1} &= {}^{VR}\mathbf{q}_t \otimes {}^{V}\mathbf{q}_{t,t+1}, \end{cases}$$

where ${}^{V}\mathbf{x}_{t,t+1}$ is the virtual state transformation, and ${}^{VR}\mathbf{q}_t$ is the quaternion of ${}^{VR}\mathbf{R}_t$. Then the iterated correction step can be formulated as an optimization problem that minimizes the deviation from the state prior, the residual derived from the real lidar measurement model, and the drift between virtual and real lidar measurements:

$$\min_{\delta\mathbf{x}, {}^{VR}\mathbf{R}_t, {}^{VR}\mathbf{p}_t} \|\delta\mathbf{x}\|_{\mathbf{\Lambda}_t} + \|{}^{R}F(\mathbf{x}^{\text{pri}}_{t,t+1} \boxplus \delta\mathbf{x})\|_{(\mathbf{J}_t\mathbf{M}_t\mathbf{J}_t^T)^{-1}} + \|{}^{VR}F(\mathbf{x}^{\text{pri}}_{t,t+1} \boxplus \delta\mathbf{x}, H(\mathbf{x}^{\text{pri}}_{t,t+1} \boxplus \delta\mathbf{x}))\|_2, \quad (4)$$

where $\|\cdot\|$ is the Mahalanobis norm (weighted Euclidean norm), $\mathbf{\Lambda}_t$ is the covariance of $\delta\mathbf{x}$, $\mathbf{J}_t$ is the Jacobian of ${}^{R}F(\cdot)$ w.r.t. the real lidar measurement noise and $\mathbf{M}_t$ is the covariance matrix of the real lidar measurement noise. The noise of virtual lidar measurement is assumed to be zero. The function ${}^{R}F(\cdot)$ measures the residual error calculated from real points ${}^{R}\mathbf{z}_t$ (determined by real state) and real edges computed based on [18]. The function ${}^{VR}F(\cdot)$ measures the drift error calculated from real points ${}^{R}\mathbf{z}_t$ and virtual points ${}^{V}\mathbf{z}_t$ (determined by virtual and real states).

The above problem can be viewed as a drift-aware VR version of the standard Kalman filter problem. The drift term makes the variable $\delta\mathbf{x}$ coupled in multiple norm terms. We propose a penalty alternating minimization (PAM) based on variable splitting. In particular, introduce a copy $\delta\mathbf{y} = \delta\mathbf{x}$ and replace $\delta\mathbf{x}$ with $\delta\mathbf{y}$ in the function ${}^{VR}F(\cdot)$. The newly introduced equality constraint is transformed into a penalty term $\rho\|\delta\mathbf{x} - \delta\mathbf{y}\|_{\mathbf{\Lambda}_t}$ ($\|\cdot\|_{\mathbf{\Lambda}_t}$ is adopted to simplify subsequent problem solving) and added to the objective function of (4), where ρ is a hyper-parameter. The penalty problem is then solved by an alternating minimization approach that optimizes $\delta\mathbf{x}$ with $\{{}^{VR}\mathbf{R}_t, {}^{VR}\mathbf{p}_t, \delta\mathbf{y}\}$ fixed, and optimizes $\{{}^{VR}\mathbf{R}_t, {}^{VR}\mathbf{p}_t, \delta\mathbf{y}\}$ with $\delta\mathbf{x}$ fixed:

$$\min_{\delta\mathbf{x}} \|\delta\mathbf{x}\|_{\mathbf{\Lambda}_t} + \|{}^{R}F(\mathbf{x}^{\text{pri}}_{t,t+1} \boxplus \delta\mathbf{x})\|_{(\mathbf{J}_t\mathbf{M}_t\mathbf{J}_t^T)^{-1}} + \rho\|\delta\mathbf{x} - \delta\mathbf{y}\|_{\mathbf{\Lambda}_t}, \quad (5a)$$

$$\min_{\delta\mathbf{y}, {}^{VR}\mathbf{R}_t, {}^{VR}\mathbf{p}_t} \|{}^{VR}F(\mathbf{x}^{\text{pri}}_{t,t+1} \boxplus \delta\mathbf{y}, H(\mathbf{x}^{\text{pri}}_{t,t+1} \boxplus \delta\mathbf{y}))\|_2 + \rho\|\delta\mathbf{x} - \delta\mathbf{y}\|_{\mathbf{\Lambda}_t}, \quad (5b)$$

Given an initial $\delta\mathbf{y}^{[0]}$, ${}^{VR}\mathbf{R}_t^{[0]}$, ${}^{VR}\mathbf{t}_t^{[0]}$, and by iteratively solving (5a)–(5b) according to [17] until convergence (in practice we can early stop the iteration), the solution $\delta\mathbf{x}^*$, ${}^{VR}\mathbf{R}_t^*$, ${}^{VR}\mathbf{p}_t^*$ to (4) is obtained. Finally, we update the covariance $\mathbf{\Lambda}_t$, posterior real state ${}^{R}\mathbf{x}^{\text{post}}_{t,t+1}$, and posterior virtual state ${}^{V}\mathbf{x}^{\text{post}}_{t,t+1} = H({}^{R}\mathbf{x}^{\text{post}}_{t,t+1})$. With error states in the local frame, we obtain ${}^{V}\mathbf{s}_t = ({}^{V}x_t, {}^{V}y_t, {}^{V}\theta_t)$ and ${}^{R}\mathbf{s}_t = ({}^{R}x_t, {}^{R}y_t, {}^{R}\theta_t)$ in the world frames of virtual and real spaces.

B. *Data Generation With Motion-Aware* DVSN

With synchronized vehicle poses in the VR spaces, the VR cameras would generate a pair of images ${}^{V}\mathbf{u}_k$ and ${}^{R}\mathbf{u}_k$. Ideally, we can directly merge ${}^{V}\mathbf{u}_k$ and ${}^{R}\mathbf{u}_k$ if their pixels are exactly aligned, However, due to practical simulation errors (e.g., inaccurate world, vehicle, camera models), there exist drifts between the virtual and real images, and we need to register them before fusion. Based on this observation, the proposed DVSN module consists of an image registration network $\mathsf{DVSN}_r(\cdot)$ and an image fusion network $\mathsf{DVSN}_f(\cdot)$, as shown in Fig. 4.

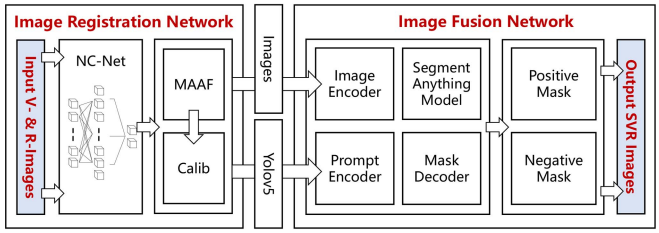


Fig. 4. Structure of DVSN for VR data generation.

For $\mathsf{DVSN}_r(\cdot)$, it utilizes the neighbourhood consensus networks (NC-Net) [19] to find a set of matching points between ${}^{V}\mathbf{u}_k$ and ${}^{R}\mathbf{u}_k$. NC-Net is a convolution neural network that identifies the above correspondences between a pair of images via analyzing neighbourhood consensus patterns in the 4D space. However, there exist incorrect matches due to the uncertainties of NC-Net.[1] These mismatched features should be discarded. Here we adopt a motion-aware adaptive filter (MAAF), which discards features by computing a time-varying region of interest (RoI) according to the vehicle motions. Specifically, this RoI is defined by a set $\mathbb{G} = \{\mathbf{g} \in \mathbb{R}^2 | \mathbf{G}\mathbf{g} \leq \mathbf{d}\}$, where $\mathbf{G} \in \mathbb{R}^{l\times 2}$ and $\mathbf{d} \in \mathbb{R}^l$, with l being the number of edges for the RoI. The center of RoI is denoted as $\mathbf{o}$. Given the actions $\{\mathbf{a}_{t+h}\}$ and the Ackerman state-action evolution model $E({}^{R}\mathbf{s}_{t+h}, \mathbf{a}_{t+h})$ (e.g., defined in (8)–(10) of [20, Sec. III-B]), the motion-aware RoI needs to minimize the total distance between model predictive trajectories and the ROI center:

$$\min_{\mathbf{o}} \sum_{h=0}^{H-1} \left\| f_{\text{proj}}({}^{R}\mathbf{s}_{t+h}) - \mathbf{o} \right\|_2^2$$
$$\text{s.t. } {}^{R}\mathbf{s}_{t+h+1} = E\left({}^{R}\mathbf{s}_{t+h}, \mathbf{a}_{t+h}\right), \ \forall h, \quad (6)$$

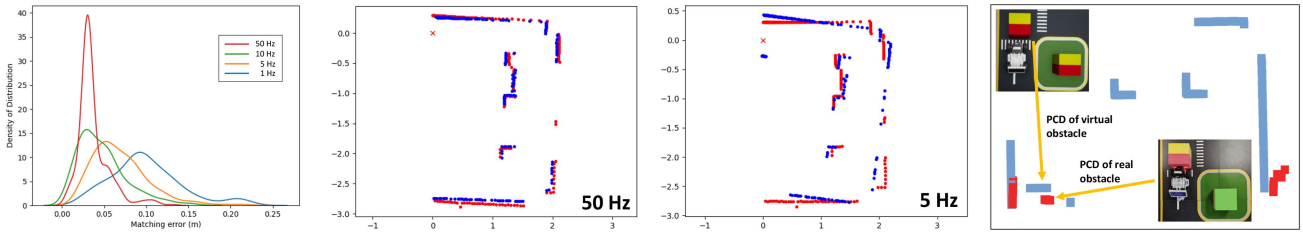
where f_{proj} is the projection from 3D space to 2D image, and H is the length of prediction horizons that can be finetuned. Denoting the solution to (6) as $\mathbf{o}^*$, the set $\mathbb{G}^*$ is obtained accordingly by varying the interior angle without changing the RoI area, e.g., reshape squares to parallelograms for $l = 4$. The MAAF would discard features outside the RoI and the pruned result is used to calculate the perspective transformation PT on virtual images. With PT, we have $\mathsf{DVSN}_r({}^{V}\mathbf{u}_k) = \mathsf{PT}({}^{V}\mathbf{u}_k)$.

Next, to merge $\mathsf{DVSN}_r({}^{V}\mathbf{u}_k)$ and ${}^{R}\mathbf{u}_k$ for data augmentation, we need to segment the useful objects out of the entire virtual image. Here we adopt the segment anything model (SAM) [21], which is a state-of-the-art large pre-trained model for image segmentation. SAM consists of image encoder, prompt encoder, and mask decoder. The image encoder and the prompt encoder

[1] Such uncertainties can be mitigated by controlling the input VR drifts below a certain threshold, which is realized by the virtual-to-real point registration in (5b).

(a) Virtual AGX (b) Real AGX (c) Virtual MoCAM (d) Real MoCAM

Fig. 5. AGX and MoCAM VR platforms



(a) Density of matching errors (b) Result at 50 Hz (c) Result at 5 Hz (d) Missed collision due to drift

Fig. 6. Quantitative and qualitative evaluations of the LIS approach

can generate image embeddings and prompt embeddings, respectively. Then the mask decoder predicts masks using prompt embeddings, image embedding, and an output token with an multi-layer perceptron for foreground probability computation. In our DVSN module, the prompt $\mathcal{P}_k$ for the image $\mathsf{DVSN}_r({}^V\mathbf{u}_k)$ is generated by bounding boxes of YOLOv5 (finetuned with the VR dataset collected from our SVR platform). Given prompt $\mathcal{P}_k$, SAM outputs negative (for object) and positive (for background) masks ${}^N\mathbf{m}_k, {}^P\mathbf{m}_k \in \{0,1\}^{LW\times 1}$, where values of 1 in ${}^N\mathbf{m}_k$ indicate pixels that belong to the object and 0 otherwise. Denoting $\odot$ as the Hadamard product for element-wise matrix multiplication, the entire procedure of DVSN can be represented as

$$\mathsf{DVSN}({}^V\mathbf{u}_k, {}^R\mathbf{u}_k) = {}^N\mathbf{m}_k \odot {}^V\mathsf{PT}(\mathbf{u}_k) + {}^P\mathbf{m}_k \odot {}^R\mathbf{u}_k. \quad (7)$$

V. Experiments

A. Implementation

We implemented the proposed SVR system using Python in ROS. The virtual world is constructed using CARLA [4], which adopts UE for high-performance rendering and PE for fine-grained dynamics modeling. The physical world is connected to CARLA via ROS bridge [13] and data sharing is realized via ROS communications, where the nodes publish or subscribe ROS topics that carry the sensory, state, or action information. The simulation worlds and IS^2 ROS packages are implemented on a Ubuntu workstation with a 3.7 GHZ AMD Ryzen 5900X CPU and an NVIDIA RTX 3090 Ti GPU. On the other hand, we use car-like robot, LIMO, as vehicles in the physical world, which adopts Ackermann steering. The robot has onboard IMU, lidar, RGBD camera, and NVIDIA Orin-Nano embedded computing chip for executing the navigation packages. A steering wheel, Logi G29, is used for recording expert driving datasets. Based on the above hardware and software, we implement two SVR platforms, i.e., the Agilex (AGX) and Macao Car Racing Metaverse (MoCAM) platforms, as shown in Fig. 5.

B. Experiment 1: Evaluation of LIS

We first evaluate the LIS module in the AGX platform. We use the VR point cloud matching error, which is the average distance between the virtual points and the associated real planes [18], as a key performance indicator to quantify the accuracy of VR colocation. The probability density of matching error under different colocation frequency is shown in Fig. 6(a). It can be seen that the matching error decreases as the colocation frequency increases. Our LIS method achieves a frequency of 50 Hz by using multi-modal measurements, and its average deviation in Fig. 6(a) is only 3 cm. This corroborates the qualitative result shown in Fig. 6(b), where the virtual and real point clouds are highly consistent. In contrast, when the frequency decreases to 5 Hz, the average deviation increases to > 5 cm. This corroborates the qualitative result shown in Fig. 6(c), where the virtual and real point clouds are now inconsistent with each other. In fact, the VR inconsistency due to low colocation frequency would become unacceptable in precrash scenarios as shown in Fig. 6(d), resulting in potential missed collisions or model clippings. This is because the ego vehicle is close to the obstacle in such cases, and a slight delay (e.g., even tens of milliseconds) between two worlds would result in a large difference in virtual and real FoVs and measurements (e.g., red points versus blue points in Fig. 6(d)).

C. Experiment 2: Evaluation of DVSN

We compare our DVSN to the following benchmarks:

- **No registration**, which directly fuses the VR images.
- **SIFT BF Matcher**, which finds and matches features with the SIFT algorithm [22] and brute-force matcher.

TABLE I
EVALUATION RESULTS OF DVSN

	R-Rate	Avg. OD	OD <5%	OD >10%
SIFT + BF matcher	47.5%	-	-	-
No Registration	-	10.9%	14.3%	45.7%
KAZE + BF Matcher	84.7%	4.9%	42.9%	5.7%
NC-Net	97.7%	3.8%	77.3%	8.6%
NC-Net with MAAF (ours)	**98.8%**	**3.2%**	**85.7%**	5.7%

- **KAZE BF Matcher**, which finds and matches features with the KAZE algorithm [23] and brute-force matcher.
- **NC-Net**, which utilizes NC-Net to identify matched feature points.

A basic filter, which solely considers the point distances, is adopted for benchmark approaches to remove outliers. The MAAF is utilized for our DVSN.

The recognizable rate (R-Rate) and object deviation (OD) are adopted to evaluate our DVSN and the four benchmark schemes. In particular, the R-Rate is defined as the number of correctly detected objects over the number of all objects [24]. This metric is used to quantify the object-level fidelity of synthesized images. On the other hand, the OD is defined as the relative position shift between the synthesized images and original images, which reflects the geometric fidelity of synthesized images and can be approximated by the mean landmark error [25]. In our experiment, we first manually label a set of matched points in each image pair as ground truth. Then we calculate the average distance between labeled points and landmarks in virtual and synthesized images, which yields d_1 and d_2, respectively. Finally, we compute OD $= |d_1 - d_2|/d_1$.

The R-Rate and OD results are summarized in Table I. The R-Rate of our approach is 98.8%, which is close to 100% and higher than all the other schemes. The average OD of our method is merely 3.2%. Furthermore, the percentage of high quality synthesized images (i.e., with OD $< 5\%$) under our approach is 85.7%, which is at least 8% higher than all the other schemes. The percentage of low quality synthesized images (i.e., with OD $> 10\%$) is 5.7%, which is also the smallest among all the evaluated schemes. Lastly, it is found that NC-Net outperforms conventional non-deep-learning registration methods, and the proposed MAAF can further improve the R-Rate performance. Note that the R-Rate of no registration scheme and the OD of the SIFT BF matcher scheme are omitted, as these metrics are not applicable to the two schemes.

D. Experiment 3: Evaluation of SVR-Trained IL

To verify the performance gain brought by SVR to AD, this experiment trains the driving model on the dataset collected in VR space and tests the SVR-trained model in real-world environments. We collect 14000 (image, action) data samples, with 7000 real samples and 7000 virtual samples, and generate another 7000 synthesized samples. The entire dataset is then categorized into the following groups:

- **Real dataset**, which only contains images and actions from the real environment (without obstacle vehicles).
- **Virtual dataset**, which only contains images and actions from the virtual environment (with obstacle vehicles).
- **No-Registration synthesized dataset**, which contains directly synthesized images and real-world actions.
- SVR **dataset**, which involves image registration and contains synthesized images and real-world actions.

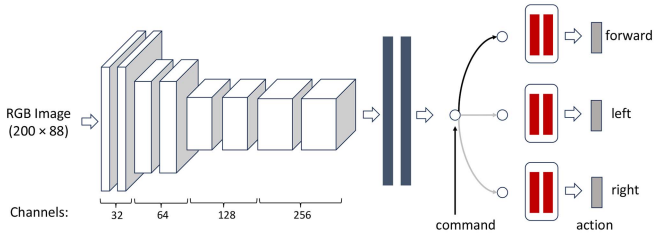


Fig. 7. Structure of the end-to-end IL network.

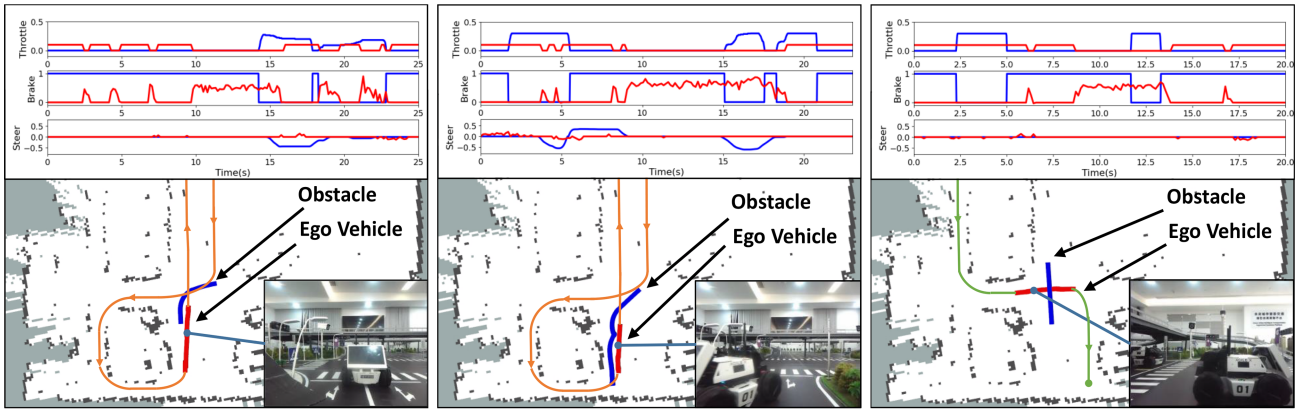
- **Enhanced** SVR **dataset**, which contains all the SVR dataset and background-changed data (<5%) generated by the simulator.

A human-in-the-loop interface is developed to allow driving expert to both see and interact with virtual content that is superimposed over the real world, which can unlock more natural human–robot interaction as shown in our video. The above datasets are separately fed to the IL network[2] shown in Fig. 7 for model training. The IL network is implemented according to [26],

Metrics and Scenarios: The number of interventions and missed-turns are used to evaluate the performance of IL models trained by different datasets [26]. In particular, we intervene the vehicle whenever it moves off the road (i.e., regarded as one intervention), and manually steer the vehicle whenever it fails to make the turn at intersections (i.e., regarded as one missed-turn). We test the trained IL models on the LIMO vehicle in the MoCAM platform and the two target routes (marked in orange and green) are shown in Fig. 8. Each route is tested 6 times. The dynamic obstacle vehicle (also LIMO), is controlled by a human expert and challenge the IL-driven vehicle according to 3 critical scenarios defined by NHTSA (https://www.nhtsa.gov/), i.e., car following, cut in, running red lights, with each scenario being tested 10 times. These challenges, together with the obstacle vehicle, are not within the real training datasets. The IL-driven vehicle is expected to accomplish the routes autonomously while avoiding collisions with any other obstacles (including both the static roads and the dynamic obstacle vehicle). Consequently, the success rate of collision avoidance is also a key metric for evaluation.

Results: The trajectories and actions of the ego and obstacle vehicles under the proposed SVR-trained IL are shown in Fig. 8, where the red trajectories are generated by the IL model and the blue trajectories are generated by the human challenger. The associated throttle, steer, and brake commands are shown in the higher part of Fig. 8. First, for the car-following scenario in Fig. 8(a), the obstacle vehicle is stopping in front of the traffic light from $t = 0$ s to $t = 14$ s. The IL-driven vehicle approaches the obstacle vehicle using a go-stop policy that generates braking signals intermittently, so as to make the best prompt reactions whenever the obstacle vehicle moves. This corroborates the activation map of IL shown in Fig. 9, which is obtained by the Grad-CAM method [27]. It can be seen that the attention layers marked in red are highly focused on the obstacle vehicle, even if the IL model has never "seen" the real LIMO before in

[2]The high-level command $c \in \{\text{forward}, \text{left}, \text{right}\}$. The output action $\mathbf{a}$ is a three-dimensional vector that controls steer s, brake b, and throttle t: $\mathbf{a} = \langle s, b, t \rangle$. Given human expert's action $\mathbf{a}_{\text{gt}} = \langle s_{\text{gt}}, b_{\text{gt}}, t_{\text{gt}} \rangle$, the training loss function is $\mathcal{L}(\mathbf{a}, \mathbf{a}_{\text{gt}}) = \lambda_s \|s - s_{\text{gt}}\|_2^2 + \lambda_b \|b - b_{\text{gt}}\|_2^2 + \lambda_t \|t - t_{\text{gt}}\|_2^2$.



(a) Trajectories and actions of ego and obstacle vehicle in car-following scenario.

(b) Trajectories and actions of ego and obstacle vehicle in cut-in scenario.

(c) Trajectories and actions of ego and obstacle vehicle in running-red-light scenario.

Fig. 8. Real-world evaluation results of SVR-trained IL in car-following, cut-in, and running-red-light scenarios.

TABLE II
COMPARISON OF IL MODELS TRAINED BY DIFFERENT DATASETS

	# intervention	# missed-turn	collision avoidance success rate			
			cut-in	car-following	running the red light	average
virtual dataset	3.33	66.7%	0.1	0.4	0.2	0.23
no-registration synthesized dataset	1	11.1%	0.5	0.6	0.5	0.53
SVR dataset (ours)	0.67	11.1%	0.6	0.7	0.7	0.67
enhanced SVR dataset (ours)	**0.33**	**5.6%**	**0.9**	**1.0**	0.7	**0.87**

Fig. 9. Activation map of IL in the car-following scenario.

the physical world. This is because the IL model learns from the VR data generated by our proposed SVR platform. As such, the navigation efficiency and safety are both guaranteed. Next, we consider the cut-in scenario shown in Fig. 8(b). Starting from $t = 0$ s, the IL-driven vehicle navigates autonomously at the right lane along the orange target path. At $t = 1.5$ s, the obstacle vehicle at the left lane accelerates to the maximum speed, turns right at $t = 3.5$ s, and after cutting in to the right lane it suddenly brakes at $t = 5.5$ s. The IL-driven vehicle reacts to the challenger properly and continues its navigation task after the challenger leaves. Finally, we consider the running-red-light scenario shown in Fig. 8(c). In this experiment, the obstacle vehicle breaks the red light at the intersection when the ego vehicle is crossing the same intersection legally. Again, the IL-driven vehicle can handle this unseen situation by leveraging its knowledge learnt from the VR space.

The quantitative results are presented in Table II. First, IL trained by virtual dataset leads to the worst performance in all metrics due to domain shifts. Compared to other schemes, its number of interventions or missed-turns is at least twice larger. In contrast, IL trained by real dataset can reduce the number of interventions or missed-turns. However, it always collides with the obstacle vehicle in the test track in our experiments (hence is omitted in Table II). These observations demonstrate the necessity of employing VR in AD. Second, by comparing the no-registration and the SVR schemes, it can be seen that ignoring registration degrades the collision avoidance capability of IL, as there exists deviation during synthesis. Third, the SVR dataset increases the success rate for collision avoidance by at least 10% compared to previously mentioned schemes in the aforementioned adversarial scenarios. This demonstrates the necessity of employing IS^2 design in VR systems. Lastly, the enhanced SVR dataset, which introduces background variation into our method would further improve the performance. This is because background variation would make the IL model more focused on foreground dynamic obstacles and less sensitive to background environments, which corroborates the observations in [28].

E. *Experiment 4: Evaluation of* SVR *on Passenger Vehicle*

Finally, we validate the robustness of SVR on a passenger autonomous vehicle in a parking lot scenario. The vehicle is equipped with 6 cameras, a 128-line 3D lidar, an RTK GPS, and an onboard Intel x86 computer. It can be seen from Fig. 10 that the trajectory (marked in blue) obtained from SVR matches the parking lot topology very well. This makes it possible for SVR to seamlessly merge virtual pedestrians into real parking lot accurately and smoothly, resulting in a naturalistic pre-crash VR testing scenario as shown in real, virtual, and SVR images at positions A and B of Fig. 10. In contrast, there exist significant deviations for the RTK-GPS trajectory (marked in red) [10]. For instance, at the upper side of the map in Fig. 10, the RTK-GPS trajectory is outside the parking lot, making the virtual vehicle crash into walls.

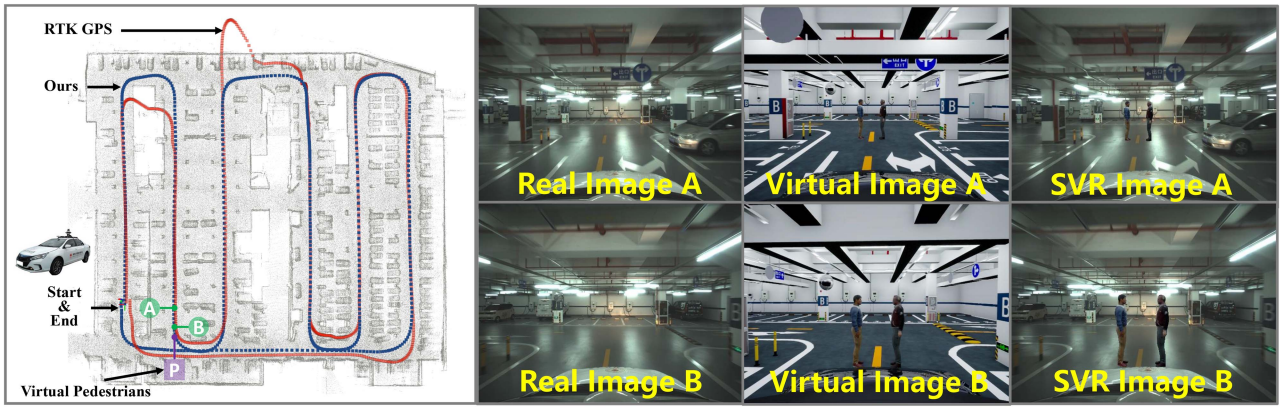


Fig. 10. Trajectory (marked in blue) and images (on the right hand side) obtained from SVR. Virtual, real, and SVR images are collected at green circles A and B, respectively. Virtual pedestrians stand at the purple square, talking with each other. For comparison, trajectory obtained from the RTK-GPS is marked in red.

VI. Conclusion

This letter presented a high-fidelity SVR platform for AD. This platform was driven by the IS^2 technique, which consists of LIS to map autonomous vehicles in the real world and adversarial obstacles in the virtual world to a symbiotic world, and DVSN to calibrate and aggregate the synchronized VR sensor and action data. Various experiments were conducted, which prove the necessity of SVR for AD and the superiority of IS^2 over other benchmarks. By integration with large multi-modal models (e.g., gemini, sora), the developed SVR platform opens up opportunities for metaverse autonomous driving.

References

[1] S. Feng, X. Yan, H. Sun, Y. Feng, and H. X. Liu, "Intelligent driving intelligence test for autonomous vehicles with naturalistic and adversarial environment," *Nature Commun.*, vol. 12, 2021, Art. no. 748.

[2] Tesla, "AI & Robotics," 2024. [Online]. Available: https://www.tesla.com/AI

[3] NVIDIA, "NVIDIA DRIVE end-to-end solutions for autonomous vehicles," 2024, [Online]. Available: https://developer.nvidia.com/drive

[4] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proc. 1st Annu. Conf. Robot Learn.*, 2017, pp. 1–16.

[5] S. Wang et al., "Federated deep learning meets autonomous vehicle perception: Design and verification," *IEEE Netw.*, vol. 37, no. 3, pp. 16–25, 2023.

[6] Y. Li et al., "V2X-Sim: Multi-agent collaborative perception dataset and benchmark for autonomous driving," *IEEE Robot. Automat. Lett.*, vol. 7, no. 4, pp. 10914–10921, Oct. 2022.

[7] W. Li et al., "AADS: Augmented autonomous driving simulation using data-driven algorithms," *Sci. Robot.*, vol. 4, no. 28, 2019, Art. no. eaaw0863.

[8] M.-Y. Liu, T. Breuel, and J. Kautz, "Unsupervised image-to-image translation networks," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 700–708.

[9] Z. Yang et al., "SurfelGAN: Synthesizing realistic sensor data for autonomous driving," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 11118–11127.

[10] S. Feng et al., "Dense reinforcement learning for safety validation of autonomous vehicles," *Nature*, vol. 615, no. 7953, pp. 620–627, 2023.

[11] S. Feng, Y. Feng, X. Yan, S. Shen, S. Xu, and H. X. Liu, "Safety assessment of highly automated driving systems in test tracks: A new framework," *Accident Anal. Prevention*, vol. 144, 2020, Art. no. 105664.

[12] Z. Zheng, X. Han, X. Xia, L. Gao, H. Xiang, and J. Ma, "OpenCDA-ROS: Enabling seamless integration of simulation and real-world cooperative driving automation," *IEEE Trans. Intell. Veh.*, vol. 8, no. 7, pp. 3775–3780, Jul. 2023.

[13] Intel, "ROS/ROS2 bridge for CARLA simulator," 2023. [Online]. Available: https://github.com/carla-simulator/ros-bridge

[14] J. Delmerico et al., "Spatial computing and intuitive interaction: Bringing mixed reality and robotics together," *IEEE Robot. Automat. Mag.*, vol. 29, no. 1, pp. 45–57, Mar. 2022.

[15] C. Li et al., "iGibson 2.0: Object-centric simulation for robot learning of everyday household tasks," in *Proc. Conf. Robot Learn.*, 2022, pp. 455–465.

[16] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "FAST-LIO2: Fast direct LiDAR-inertial odometry," *IEEE Trans. Robot.*, vol. 38, no. 4, pp. 2053–2073, Aug. 2022.

[17] M. Bloesch, M. Burri, S. Omari, M. Hutter, and R. Siegwart, "Iterated extended Kalman filter based visual-inertial odometry using direct photometric feedback," *Int. J. Robot. Res.*, vol. 36, no. 10, pp. 1053–1072, 2017.

[18] J. Zhang and S. Singh, "LOAM: Lidar odometry and mapping in real-time," in *Proc. Robot.: Sci. Syst.*, Berkeley, CA, 2014, pp. 1–9.

[19] I. Rocco, M. Cimpoi, R. Arandjelović, A. Torii, T. Pajdla, and J. Sivic, "Neighbourhood consensus networks," in *Proc. 32nd Conf. Neural Inf. Process. Syst.*, 2018, pp. 1658–1669.

[20] R. Han et al., "RDA: An accelerated collision-free motion planner for autonomous navigation in cluttered environments," *IEEE Robot. Automat. Lett.*, no. 3, pp. 1715–1722, Mar. 2023.

[21] A. Kirillov et al., "Segment anything," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.* 2023, pp. 3992–4003.

[22] D. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2012.

[23] P. F. Alcantarilla, A. Bartoli, and A. J. Davison, "KAZE features," in *Eur. Conf. Comput. Vis.*, 2012, pp. 214–227.

[24] S. Manivasagam et al., "LidarSIM: Realistic lidar simulation by leveraging the real world," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 11167–11176.

[25] J. H. Song, "Methods for evaluating image registration," Ph.D. dissertation, University of Iowa, Iowa City, IA, USA, 2017.

[26] F. Codevilla, M. Müller, A. López, V. Koltun, and A. Dosovitskiy, "End-to-end driving via conditional imitation learning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 4693–4700.

[27] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," in *Proc. IEEE Int. Conf. Computer Vis.*, 2017, pp. 618–626.

[28] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 23–30.