

# VISTA 2.0: An Open, Data-driven Simulator for Multimodal Sensing and Policy Learning for Autonomous Vehicles

Alexander Amini<sup>1,\*</sup>, Tsun-Hsuan Wang<sup>1,\*</sup>, Igor Gilitschenski<sup>2</sup>, Wilko Schwarting<sup>1</sup>, Zhijian Liu<sup>1</sup>, Song Han<sup>1</sup>, Sertac Karaman<sup>1</sup>, and Daniela Rus<sup>1</sup>

**Abstract**—Simulation has the potential to transform the development of robust algorithms for mobile agents deployed in safety-critical scenarios. However, the poor photorealism and lack of diverse sensor modalities of existing simulation engines remain key hurdles towards realizing this potential. Here, we present *VISTA*, an open source, data-driven simulator that integrates multiple types of sensors for autonomous vehicles. Using high fidelity, real-world datasets, *VISTA* represents and simulates RGB cameras, 3D LiDAR, and event-based cameras, enabling the rapid generation of novel viewpoints in simulation and thereby enriching the data available for policy learning with corner cases that are difficult to capture in the physical world. Using *VISTA*, we demonstrate the ability to train and test perception-to-control policies across each of the sensor types and showcase the power of this approach via deployment on a full scale autonomous vehicle. The policies learned in *VISTA* exhibit sim-to-real transfer without modification and greater robustness than those trained exclusively on real-world data.

## I. INTRODUCTION

Simulation has emerged as an essential tool for advancing new algorithms in robot perception, learning, and evaluation [1]–[3]. For safety-critical domains in particular, such as for autonomous vehicles, experience in simulation is often significantly faster and safer than direct operation in the physical world. Simulation affords the potential to rapidly synthesize novel data for training, including challenging edge cases difficult to capture in the real world [1], [2]. An agent’s exposure to edge cases during training is critical to achieving robustness to out-of-distribution events. Furthermore, high-fidelity, in-simulation testing could improve an agent’s performance when deployed into safety-critical, human-centric environments. Thus, simulation could enable the development of algorithms and models better equipped to handle the diverse challenges of the physical world, facilitating their deployment on embodied mobile agents.

Despite the potential of simulation, the stark lack of photorealism and a paucity of diverse simulated sensor representations have remained crucial barriers towards realizing this promise. Data-driven simulation, unlike traditional model-based simulation, synthesizes novel viewpoints directly from real data and has emerged as an approach to overcome the photorealism and sim-to-real gap [4]. However, issues

\* The first two authors have contributed equally to this work. This work was supported by National Science Foundation and Toyota Research Institute. We gratefully acknowledge the support of NVIDIA with the donation of the Drive AGX Pegasus.

<sup>1</sup> Department of Electrical Engineering and Computer Science (EECS), Massachusetts Institute of Technology (MIT). {amini, tsunw, wilkos, zhijian, song, sertac, rus}@mit.edu

<sup>2</sup> Department of Computer Science, University of Toronto and Toyota Research Institute (TRI). gilitschenski@cs.utoronto.edu

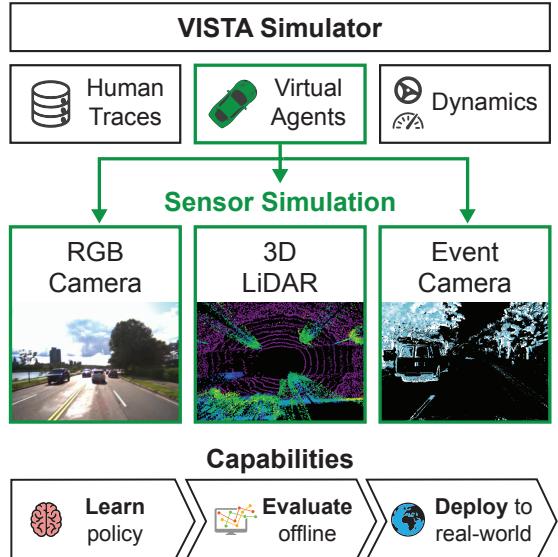


Fig. 1: *VISTA 2.0* is an open-source data-driven simulator for multi-sensor perception of embodied agents. Leveraging data of the real-world, *VISTA* synthesizes ego-agent viewpoints as their dynamics unroll novel trajectories in the environment. Sensors are efficient and high fidelity for online perception learning, evaluation, and sim-to-real deployment. In scaling simulation engines to multiple sensor types for online perception learning remain. Because embodied agents benefit from rich perception [5], integrating multiple sensor modalities could facilitate adaptation to a wide variety of environmental conditions (e.g., combining LiDAR and camera feedback to stay on the road in low visibility lighting). There remains a need for unified, flexible, and open-source data-driven simulation engines to fuel the development of new algorithms for embodied agent learning and evaluation.

In this paper, we present *VISTA 2.0*, a multi-sensor, data-driven engine for autonomous vehicle simulation, perception, and learning (Fig. 1). *VISTA* synthesizes novel viewpoints consistent with each sensor representation, simulates agents in synthesized scenarios, and supports large scale learning and testing environments. Our work prioritizes a lightweight API for processing existing real-world datasets, operating only on local viewpoint changes to achieve efficient computational rendering and low memory costs.

We develop novel view synthesis capabilities for three distinct sensors: 2D RGB cameras, 3D LiDARs, and asynchronous event-based sensors. Using *VISTA* to generate data, we train end-to-end (i.e., perception-to-control) policies using guided policy learning (GPL) [6] and demonstrate

direct policy transfer onto a full-scale autonomous vehicle. Further, our results highlight the importance of simulation in two key contexts. First, we show that simulating novel viewpoints can drastically improve the robustness of learned policies and a vehicle’s ability to recover from challenging off-orientation positions. Second, for sensors which conflate the ego-motion of the agent with the control decision (e.g., event-based cameras), we find that existing state-of-the-art imitation learning (IL) approaches [7] cannot achieve closed-loop success using only real-world data. Using **VISTA**, we not only uncover this issue, but also overcome it by decoupling the learning signal within **VISTA** to successfully train closed-loop controllers from event-based sensors.

In summary, the contributions of this paper are as follows:

- 1) **VISTA 2.0**, an open-source, multi-sensor, data-driven simulator for learning and evaluating autonomous vehicle perception and control.
- 2) A framework for translating real-world data to a simulated perception-control API spanning a diversity of compatible environments with varying complexity, lighting, weather, and road types.
- 3) End-to-end autonomous vehicle control policies using each sensor type, learned within **VISTA** and directly deployed on a full-scale vehicle. Learned policies exhibit direct sim-to-real transfer and improved robustness than those trained solely on real-world data.

## II. RELATED WORK

**Cross-Sensor Transfer:** Numerous works consider augmenting sensing modalities and simulating different modalities via, often learned, sensor fusion. For example, in *Monocular Depth Prediction* approaches, a neural network is typically trained to act as a depth sensor using monocular image data [8], [9]. Similarly, *Depth Completion* combines cameras with sparse depth from LiDAR to simulate a dense depth sensor [10], [11]. In event-based vision, recent work combine the use of classical and event-based cameras for simulating higher frame-rate cameras [12], depth predictions [13], [14], or focus on translating between these two modalities [15]–[18]. We draw inspiration from these works but focus on a unified simulation framework that jointly simulates a diverse set of sensors while supporting novel view synthesis.

**Simulation:** The use of simulation for learning and robotics has exploded in recent years. Model-driven simulators rely on predefined models of scenery and underlying physics [19]–[23]. Several model-based engines that focus on high quality visual appearances are widely used for autonomous vehicles [1], [24] and drones [25], [26]. These engines rely on heavily engineered video-game rendering platforms, but still lack the photorealism necessary for direct policy transferability. In contrast, data-driven simulators [2], [4], [27]–[30] present greater photorealism by leveraging real data to reconstruct virtual worlds of the scene before synthesizing novel views. Our work follows this line of research with a focus on local scene synthesis for scalability and on learning transferable policies for embodied AI research.

**Driving Policy Learning:** While policy learning for driving using real-world data is largely restricted to IL [31]–[35], learning in simulation allows for greater algorithmic

flexibility ranging from IL [5], [36], [37], to RL [1], [4], [38], [39], and GPL [6], [40]. Evaluation of trained policies in closed-loop simulation [5], [31], [36], [40]–[42] also presents benefits over open-loop evaluation [7], [32], [43]. Similarly, our work leverages simulation for edge-case training data generation, and closed-loop evaluation before deployment.

## III. MULTI-SENSOR SIMULATION

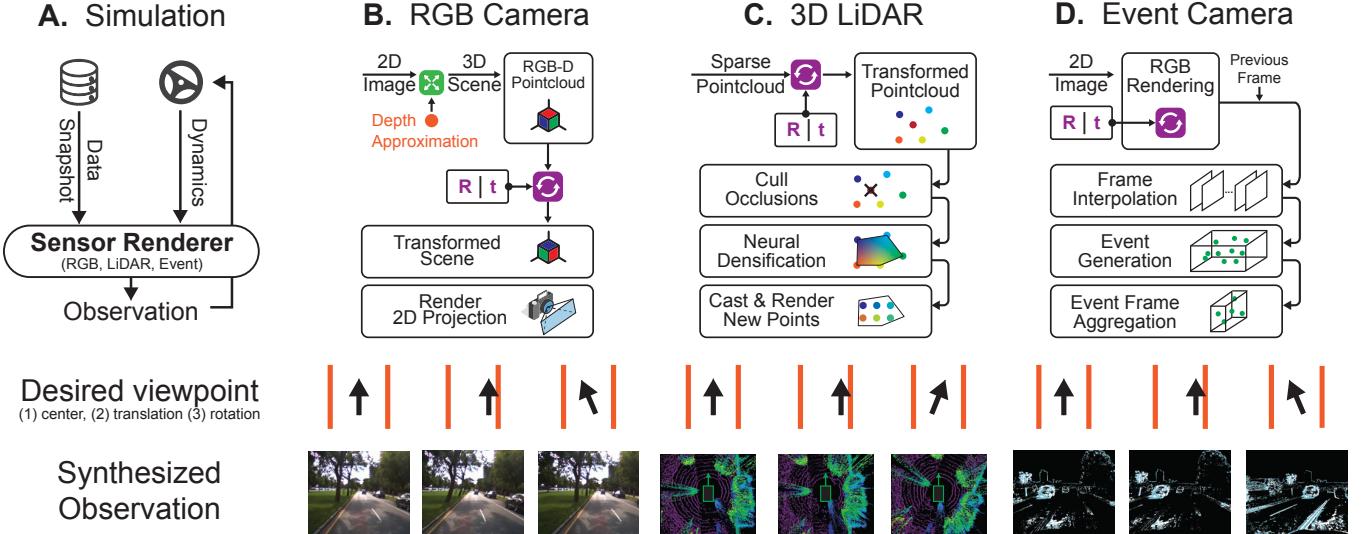
### A. Background

**VISTA 1.0** [4] is a data-driven simulator that synthesizes RGB images at novel viewpoints around the local trajectory of a dataset. The precollected image sequence represents a sparsely sampled representation of a continuous trajectory traversed by a vehicle in the physical world. Any novel viewpoints can be associated with a frame with the closest pose and rerendered to the virtual agent’s position. The overall pipeline of **VISTA** is: (1) update vehicle state with a continuous kinematic model, (2) retrieve the closest frame in the dataset with respect to the current pose, (3) project the frame into 3D space to reconstruct the scene, and (4) reproject back into the ego-agent’s point-of-view. Please refer to [4] for details. The goal of **VISTA 2.0** is to extend simulation to other modalities in a data-driven manner, namely synthesizing novel sensor measurements of LiDAR and event data locally around the dataset, and to leverage and release this platform for robust perception learning.

### B. LiDAR synthesis

LiDAR sensors play a central role in modern autonomy pipelines due to their accuracy in measuring geometric depth information and robustness to environmental changes like illumination. Unlike cameras which return structured grid-like images, the LiDAR sensor captures a sparse pointcloud of the environment. Here, every point is represented by a 4-tuple:  $(x, y, z, i)$ , where  $(x, y, z)$  is the position of the point in 3D Cartesian space and  $i$  is the intensity feature measurement of that point. Given a virtual agent’s position in the environment, along with a relative transformation (rotation  $R \in \mathbb{R}^{3 \times 3}$ , and translation  $t \in \mathbb{R}^3$ ) to the nearest human collected pointcloud,  $\Psi$ , the goal of **VISTA** is to synthesize a novel LiDAR pointcloud,  $\Psi'$ , which appears to originate from the virtual agent’s relative position.

Since  $\Psi$  is represented in 3D Cartesian space, a naive solution would be to directly apply the relative transformation of the agent ( $R, t$ ) to  $\Psi$  as a rigid transformation:  $\Psi' = R\Psi + t$ . However, this approach will fail for several reasons. The pointcloud obtained from a LiDAR sensor has a specific ring pattern pattern originating at the sensor’s optical center. Applying a rigid transformation to the points will not only transform the individual points, but in doing so, also transform and break the ring structure inherently defining the sensor’s location. Instead, to preserve the sensor structure we must recast LiDAR rays, from the new sensor location, into the scene and estimate new readings. Furthermore, the naively transformed pointcloud will very likely have points which may have been visible in the original scan, but become occluded in the new viewpoint and thus need to be rejected to maintain line-of-sight properties of the sensor. To overcome these issues, we (1) cull the now occluded points, (2) create a



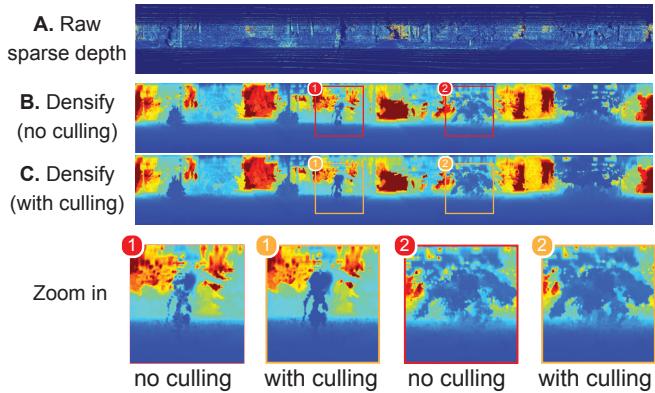
**Fig. 2: Multi-sensor simulation.** (A) Virtual agent dynamics unroll trajectories in a data-driven environment, at each time rendering a sensor observation from the novel viewpoint at that time. Three types of perception sensors are able to be synthesized including images from RGB cameras (B), 3D pointclouds from LiDAR (C), and continuous differential events (D). Examples of novel viewpoint synthesis are visualized for each sensor.

dense representation of the sparse pointcloud, and (3) sample from the dense representation according to a sensor-specific prior. We outline the algorithm below in detail.

First, we implement a GPU-accelerated culling technique to operate on our sparse transformed pointcloud,  $\Psi'$ . We start by projecting  $\Psi'$  into 2D polar coordinates,

$$\alpha = \arctan\left(\frac{\Psi'_y}{\Psi'_x}\right); \quad \beta = \arcsin\left(\frac{\Psi'_z}{d}\right); \quad d = \|\Psi'\|_2 \quad (1)$$

where  $(\alpha, \beta)$  are the yaw and pitch angles of the rays connecting each of the points, and  $d$  are the distances along each ray. Now, the entire pointcloud,  $\Psi'$  is represented as a sparse 2D image (without loss of information) over  $(\alpha, \beta)$  with  $d$  being the color or value of each pixel. To cull out points within our image, the distance of each pixel is compared to the average distance of its surrounding “cone”

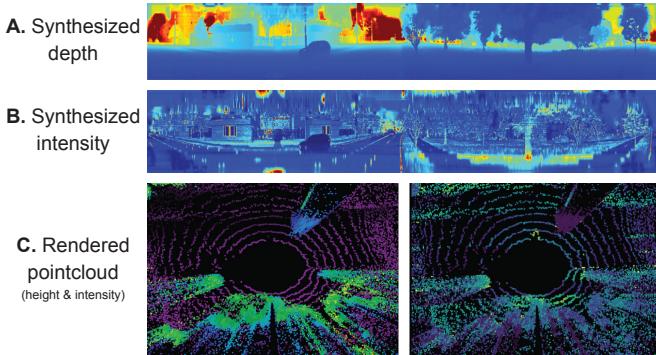


**Fig. 3: Culling occluded points.** Transformed sparse scenes (A) will have points which should be rejected (culled) before rendering to avoid blending of foreground and background (B). Our culling algorithm (C) is lightweight, GPU-accelerated, and does not rely on raycasting a scene mesh.

of neighboring rays. If the average distance of neighboring rays is less than the depth of the current pixel, the point is occluded and is removed from the sparse image. Fig. 3 visualizes the large effect of our culling algorithm and the qualitative improvement it has on transformed pointclouds.

With our sparse and culled pointcloud, we need to build a dense representation of the scene to sample a new cast of LiDAR rays and generate the novel viewpoint. To densify our sparse representation we train a UNet architecture [44] to learn a dense output of the scene. Training data for our densification network is generated using a 2D linear interpolator. We found that using a data-driven approach to densification yielded smoother, more natural qualitative results over strict rule-based interpolation (`scipy.interpolate`). Furthermore, the resulting model is easily GPU-parallelizable to achieve significant speedups ( $\sim 100\times$  faster).

Finally, we sample sparse points from our dense representation to form the novel view pointcloud. To determine sampling locations we can construct a prior,  $\Omega$ , over the existing ray cast angles of the sensor in our dataset. The ray vectors for the sensor are largely fixed over time, as they are built into the hardware of the sensor, but can have some slight variations or drops based on the environment. Sampling  $\omega$  from the prior yields a collection of rays,  $\{(\alpha_i, \beta_i)\}$ , to cast and collect point readings from. Furthermore, the prior,  $\Omega$ , will respect several desirable properties of the sensor which can also be user specified such as the quantity and density of the LiDAR rays. Since we are still operating in polar coordinate image space,  $\omega$  is equivalent to a binary mask image denoting where in our dense image should be sampled. With our new, sampled polar image we can invert the transform in Eq. 1 to represent our data back in the desired 3D Cartesian space. Fig. 4 visualizes the different stages in the rendering pipeline, through the dense representation of the scene (A,B) as well as the result after sampling and reprojecting back to 3D cartesian space (C).



**Fig. 4: LiDAR novel view-synthesis.** Simulating a lateral translation of 1m off the road. Dense representations of depth (A) and intensity (B) are estimated from the sparse transformation. Sparse pointclouds (C) are rendered by sampling the dense representation according to the sensor prior.

### C. Event synthesis

Event-based cameras are asynchronous, continuous-time sensors that detect brightness changes of the scene. An event is emitted when brightness change exceeds a certain threshold at a pixel location, and is described as a 4-tuple of pixel coordinate, timestamp, and polarity. The polarity is a binary value that indicates whether brightness change is positive or negative. Conceptually, event camera can be viewed as the derivative of regular RGB camera with additional advantages of much higher operating frequency ( $> 10,000\text{Hz}$ ) and dynamic range. Given its similarity to RGB camera, event data can be simulated by taking derivative with respect to time over interpolated RGB frames [15], [45]. Our proposed method extends prior work to additionally handle (1) non-aligned camera projection across RGB and event camera and (2) novel view synthesis according to vehicle’s ego-motion. Simulating events from RGB instead of event data allows applying **VISTA** to existing datasets which mostly contain RGB sequences but not event data.

To capture the instantaneous change of pixel intensity, we need to first construct a continuous representation of RGB image stream. Given two consecutive RGB frames  $I_{t_1}, I_{t_2}$  and bidirectional optical flow  $F_{t_1 \rightarrow t_2}, F_{t_2 \rightarrow t_1}$ , this can be achieved by arbitrary-time frame interpolation [46],

$$I_{t_1+k\Delta t} = f_{\text{interp}}(I_{t_1}, I_{t_2}, F_{t_1+k\Delta t \rightarrow t_1}, F_{t_1+k\Delta t \rightarrow t_2}) \quad (2)$$

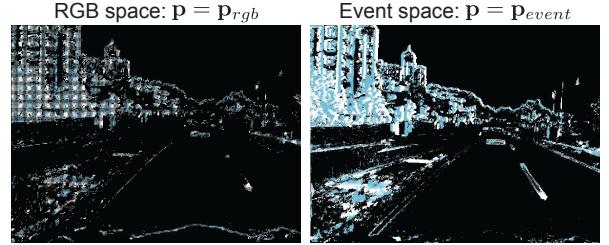
$$F_{t_1+k\Delta t \rightarrow t_1} = -(1-k\Delta t)k\Delta t F_{t_1 \rightarrow t_2} + (k\Delta t)^2 F_{t_2 \rightarrow t_1} \quad (3)$$

$$F_{t_1+k\Delta t \rightarrow t_2} = (1-k\Delta t)^2 F_{t_1 \rightarrow t_2} - k\Delta t(1-k\Delta t) F_{t_2 \rightarrow t_1} \quad (4)$$

where  $k \in [0, \frac{t_2-t_1}{\Delta t}]$  and  $k\Delta t$  specifies the time interval to be simulated in between. The arbitrary-time flow is derived from temporal consistency going forward ( $t_1 \rightarrow t_1 + k\Delta t$ ) and backward ( $t_2 \rightarrow t_1 + k\Delta t$ ) in time with local smoothness assumption. The interpolation function  $f_{\text{interp}}$  is implemented by a neural network that handles visibility issue from both directions. We refer the reader to [46] for more details. With this, we now apply an event generation model [47], [48],

$$(\mathbf{p}, t_k, \rho) \text{ if } \rho (\ln I(\mathbf{p}, t_k) - \ln I(\mathbf{p}, t_k - \Delta t)) \geq c_k \quad (5)$$

where  $\mathbf{p}$  is the pixel coordinate,  $\rho \in \{-1, 1\}$  is polarity and  $c_k \sim \mathcal{N}(\mu_c, \sigma_c)$  is the contrast threshold sampled from a



**Fig. 5: Different pixel spaces for event generation.** Events can be generated by estimating brightness change in RGB camera image space or virtual event camera image space.

Gaussian to simulate noise. The temporal granularity of event generation is determined by  $\Delta t$ , which yields more accurate simulation with smaller values until saturating at subpixel displacement of optical flow. Furthermore, adaptive sampling [15] is used to jointly achieve accuracy and efficiency,

$$\Delta t = \frac{t_2 - t_1}{\min\{\max\max\{F_{t_1 \rightarrow t_2}(\mathbf{p}), F_{t_2 \rightarrow t_1}(\mathbf{p})\} - 1, \Delta t_{\max}\}} \quad (6)$$

where  $\Delta t_{\max}$  sets an upper bound to computational resources required for the simulation. Thus far, the only thing yet to be defined is the space that pixel coordinate  $\mathbf{p}$  lives in. It can be image space of either RGB camera (input) or novel-view event camera (output), which are related by reprojection,

$$\mathbf{p}_{\text{event}} = K_{\text{event}} T_{\text{event}}^{novel} T_{\text{rgb}}^{\text{event}} D(\mathbf{p}_{\text{rgb}}) K_{\text{rgb}}^{-1} \mathbf{p}_{\text{rgb}} \quad (7)$$

where  $K_{\text{event}}, K_{\text{rgb}}$  are intrinsics for event and RGB cameras,  $D$  is depth, and  $T$  is transformation across two poses. We explicitly factorize  $T_{\text{rgb}}^{\text{event}}$  since the control commands reference at existing sensors in the dataset. In event generation model (5), using  $\mathbf{p}_{\text{rgb}}$  involves generating events in RGB image space from the dataset and reprojecting pixel coordinates to event image space. We implement a bilinear sampler with thresholding to handle non-integer pixels after reprojection. On the other hand, using  $\mathbf{p}_{\text{event}}$  renders the scene based on RGB dataset in event camera space and generate events without pixel reprojection. We argue that  $\mathbf{p} = \mathbf{p}_{\text{event}}$  may be a better option since it casts the subpixel issue of reprojection (7) from interpolating in pixel coordinate space as in  $\mathbf{p} = \mathbf{p}_{\text{rgb}}$  to interpolating in color/intensity space of meshes during RGB image rendering. The comparison can be seen in Fig. 5.

## IV. POLICY LEARNING

### A. Data generation and training using VISTA

The capability of **VISTA** to simulate sensor measurements allows for reinforcement learning [4] as well as guided policy learning by generation of novel view training data. With access to the internal state of the simulator, a controller with privileged information [6], [31], [40] (GPL) can generate optimal control commands corresponding to agent’s current state. Associating this control with perception data increases diversity of training data distribution and allows generating edge cases locally around the dataset, thus improving the recovery robustness of the policy. Algorithm 1 describes data generation in **VISTA** and GPL. First, the simulator is reset with random initialization (e.g., off-center position and heading). The simulator is stepped with control commands

from the privileged controller and iteratively generates sensor measurements and optimal control labels for supervising policy learning. While there are no restrictions on the optimal controller used for the privileged agent, we use a pure-pursuit controller in our experiments. We use a shuffled buffer to approximately ensure *i.i.d.* training samples. Before adding data into the buffer, rejection sampling is used to balance the label distributions [33]. Finally, we introduce a branching step that locally branches out from stepping the simulator with privileged control (e.g., turning right instead of left as instructed). This is extremely important to policy learning with event cameras since event data captures *changes in the scene* and thus conflates the vehicle’s ego-motion with its own control. This means that a policy can accurately correlate control to the motion of the scene instead of attending to the road. While not presenting an obvious issue during open-loop evaluation [7], these policies will fail catastrophically on a closed-loop test. However, by branching with arbitrary control, we effectively disentangle ego-motion and scene information in event patterns.

#### Algorithm 1 Data generation and training in VISTA

```

for  $k \leftarrow 1$  to  $N$  do
    while ! buffer.full() do
        if VISTA.done() then
            VISTA.reset()
        end if
         $x \leftarrow$  VISTA.readSensorBuffer()
        if useBranching then
            VISTA.randomStep()
             $y \leftarrow$  privilegedController(VISTA.getState())
            revertState(VISTA, privilegedController)
        else
             $y \leftarrow$  privilegedController(VISTA.getState())
        end if
        VISTA.step( $y$ )
        if ! rejectSample( $x, y$ ) then
            buffer.add( $x, y$ )
        end if
    end while
    buffer.shuffle()
    trainModel(buffer.next())
end for

```

#### B. Input representation and model architectures

All models consist of a feature extractor which processes the sensory data and a deterministic estimator which learns control from the features. We use the same network architecture for the estimator (a 3-layer fully connected network), which outputs a scalar value as curvature. For RGB images, we use a simple 5-layer CNN, each layer comprises a convolution, group norm, and ReLU [33]. For event data, we accumulate events within a small time interval, project them on a frame according to their pixel coordinates and apply a convolutional feature extractor [7]. For LiDAR data, we use FastLiDARNet [49], [50] that can process point cloud efficiently with sparse tensor operation.

## V. RESULTS

#### A. Experimental setup and data collection

**Hardware setup.** We collect data and deploy learned policies on a full-scale vehicle (2019 Lexus RX 450H) which we have outfitted with autonomous driving capabilities. The

car is equipped with an NVIDIA 2080Ti GPU and an AMD Ryzen 7 3800X 8-Core Processor. Perception sensors include a 30Hz BFS-PGE-23S3C-CS RGB camera , a 10Hz Velodyne VLS-128 LiDAR sensor, and a Prophesee Gen3 event-based camera. The event camera runs at adaptive rate based on events emission which range from hundreds to thousands Hz. Other on-board sensors include inertial measurement units (IMUs) and wheel encoders for estimating odometry as well as a centimeter-level accurate OxTS global positioning system (d-GPS) for evaluation.

**Data collection.** We collect data from multiple sensors (RGB camera, LiDAR, event camera) in a wide variety of environments, including different time of day (daytime/night), weather conditions (sun/rain), and road types (urban/rural). The entire dataset contains roughly 3 hours of driving data. RGB images, LiDAR point cloud, event data, and curvature feedback are used for **VISTA** simulation, policy learning and evaluation. GPS data is only used for evaluation.

**Evaluation metrics.** For open-loop tests, we compute mean squared errors between human control command (curvature) and policies’ predictions. For closed-loop tests, we compute mean deviation from the lane center, crash rate, and number of interventions. In simulation, we consider crashes as lateral translation from human trajectories larger than 2 meters. In real-world tests, we intervene and take over control from autonomous mode once vehicle is off the road.

#### B. Offline evaluation

In Tab. I, we show open-loop control errors of IL and GPL. Note that the error is of the scale  $10^{-5}$  (normal driving roughly has curvature between  $\pm 0.05$ ) and thus both algorithms perform quite well in spite of the difference. IL outperforms GPL as expected given IL’s training objective is aligned with such evaluation. However, open-loop error is a very poor indicator of measuring the performance of a driving policy [41] since it (1) only measures errors around human trajectories and (2) ignores compounding feedback errors that bring the vehicle to out-of-distribution states.

Instead, closed-loop test settings (either in simulation or reality) serve as a greater proxy for evaluating policy performance. We start by using **VISTA** to measure closed-loop performance with synthetic data before moving to the real-world. Tab. II shows closed-loop performance of policies in **VISTA**. Consistent with prior research, we qualitatively observe a common failure mode of IL policies where they gradually drift off the road and cannot recover (due

Sensor	Algo.	Mean Squared Error (1E-5)					
		Day	Night	Sun	Rain	Urban	Rural
RGB	IL	0.64	0.15	0.51	0.39	0.69	2.10
	GPL	3.01	0.61	22.44	2.34	14.92	4.75
LiDAR	IL	0.92	1.04	0.74	2.52	1.02	0.61
	GPL	7.35	9.83	8.28	11.12	8.87	5.07
Event	IL	0.18	0.26	0.37	1.24	0.19	10.81
	GPL	20.75	10.12	16.54	6.54	10.88	10.25

Tab. I: **Open-loop control errors.** Open-loop IL outperforms GPL when considering only error. Note that low open-loop error is a very poor indicator for closed-loop success [41].

Sensor	Algo.	Mean Deviation						Crash Rate					
		Day	Night	Sun	Rain	Urban	Rural	Day	Night	Sun	Rain	Urban	Rural
RGB	IL	0.283	0.165	0.289	0.425	0.285	0.191	0.080	0.008	0.090	0.010	0.146	0.094
	GPL	0.102	0.068	0.120	0.101	0.120	0.226	0.002	0.000	0.002	0.006	0.004	0.000
LiDAR	IL	0.327	0.302	0.295	0.366	0.307	0.213	0.664	0.656	0.652	0.734	0.668	0.330
	GPL	0.266	0.258	0.280	0.323	0.268	0.274	0.334	0.330	0.322	0.426	0.316	0.150
Event	IL	0.340	0.344	0.327	0.320	0.329	0.362	0.486	0.828	0.714	0.576	0.674	0.784
	GPL	0.307	0.313	0.293	0.278	0.319	0.376	0.166	0.200	0.084	0.051	0.324	0.442

Tab. II: **Closed-loop performance of policies in VISTA.** GPL policies exhibit consistently reduced deviation from the center line than compared to real-world IL. Furthermore, these polices exhibit greater robustness with lower crash rates.

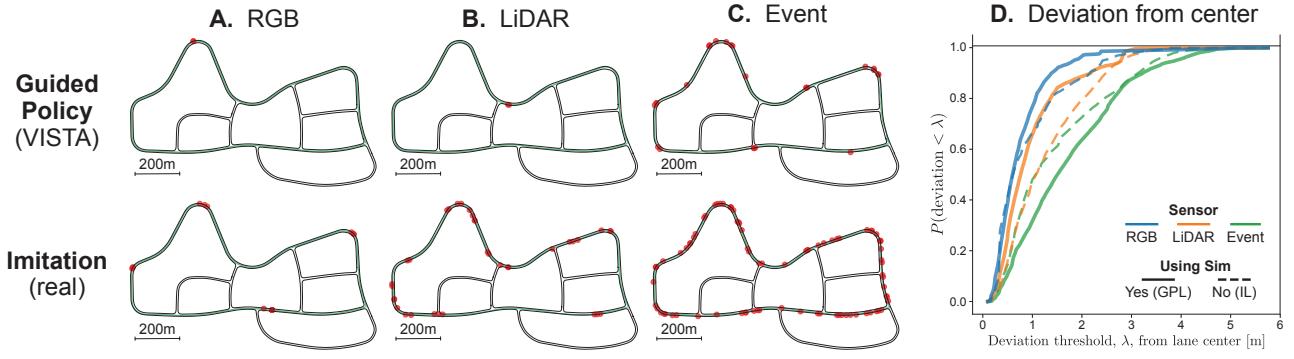


Fig. 6: **Real-world deployment.** Policy trajectories ( $n=3$ ; 45km total) and crash locations (red dots) for RGB (A), LiDAR (B), and Event (C) sensors. Cumulative distribution of deviations from the center (D) shows the benefit of training in **VISTA**.

to lack of recovery training data). GPL policies exhibit consistently reduced deviation from the lane center and lower crash rate than IL policies in a wide range of environments, lighting, and sensors modalities.

### C. Online real-world test

In Fig. 6, we demonstrate real-world policy deployment of IL and GPL policies. For each policy, we run the vehicle autonomously (controlled by the policy) for 3 trials in the outerloop of the test track (total distance of all trials is 45km). Fig. 6(A-C) show interventions (red dots) throughout multiple trials. Fig. 6(D) shows percentage of deviation from center smaller than a range of thresholds, where larger area below the line means more stable lane keeping maneuvers. The performance of GPL policies for RGB and LiDAR are significantly better than IL policies. This is highly aligned with our observation from closed-loop testing in **VISTA**(Tab. II), which further motivates its effectiveness for policy evaluation, considering the time and safety costs of real-world testing. For event camera, while GPL policy also exhibit superior performance in terms of number of interventions, it deviates more from the center compared to IL policy and suffers from much frequent intervention compared to RGB and LiDAR GPL policies.

Recovery Rate	RGB	LiDAR	Event
IL	$0.27 \pm 0.13$	$0.16 \pm 0.11$	$0.00 \pm 0.00$
GPL	$0.90 \pm 0.13$	$0.83 \pm 0.22$	$0.90 \pm 0.15$

Tab. III: **Robustness Test.** GPL policies significantly outperform real-world IL at recovering from edge cases.

This is due to the fact that event cameras can only see the component of the road boundary non-parallel to vehicle’s ego motion, while RGB and LiDAR sensors provide sufficient information at every step for lane following. Such properties highlight the potential utility of fusing event sensing with RGB and LiDAR for greater benefits. We observed a swirling maneuver along straight roads with event policies (8x more jittery than RGB (measured by squared second derivative of curvature)). However, we found that our method of branched learning for preventing ego-motion conflating the scene understanding (Sec. IV-A) is highly effective in real-world tests to reduce the number of interventions:  $28.0 \pm 3.6$  (IL)  $18.0 \pm 1.0$  (GPL)  $6.5 \pm 3.8$  (GPL with branching). To further highlight the effectiveness of GPL policies, we conduct a robustness test by initializing the car with  $\pm 30^\circ$  rotation and  $\pm 2m$  translation from the lane center and measure the success rate of recovery, as shown in Tab. III.

## VI. CONCLUSION

We present **VISTA**, an open-source simulator that supports multimodal sensor synthesis including 2D RGB cameras, 3D LiDAR, and event-based cameras for mobile agents. The simulator is data-driven and capable of synthesizing high-fidelity sensor measurement sufficient for policy learning and evaluation. We showcase the sim-to-real ability by directly deploying policies learned in **VISTA** on a full-scale autonomous vehicle for each sensor and demonstrate consistent results between closed-loop evaluation in simulation and real-world test. We believe the release and scalability of **VISTA** opens up new research opportunities to the community for perception and control of autonomous vehicles.

## REFERENCES

- [1] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, “CARLA: An Open Urban Driving Simulator,” in *CoRL*, 2017.
- [2] M. Deitke, W. Han, A. Herrasti, A. Kembhavi, E. Kolve, R. Mottaghi, J. Salvador, D. Schwenk, E. VanderBilt, M. Wallingford, L. Weih, M. Yatskar, and A. Farhadi, “RoboTHOR: An Open Simulation-to-Real Embodied AI Platform,” in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- [3] S. Shah, D. Dey, C. Lovett, and A. Kapoor, “AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles,” in *FSR*, 2017.
- [4] A. Amini, I. Gilitschenski, J. Phillips, J. Moseyko, R. Banerjee, S. Karaman, and D. Rus, “Learning robust control policies for end-to-end autonomous driving from data-driven simulation,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1143–1150, 2020.
- [5] Y. Xiao, F. Codevilla, A. Gurrum, O. Urfalioglu, and A. M. López, “Multimodal End-to-End Autonomous Driving,” *arXiv:1906.03199*, 2019.
- [6] S. Levine and V. Koltun, “Guided policy search,” in *International conference on machine learning*. PMLR, 2013, pp. 1–9.
- [7] A. I. Maqueda, A. Loquercio, G. Gallego, N. García, and D. Scaramuzza, “Event-based vision meets deep learning on steering prediction for self-driving cars,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5419–5427.
- [8] C. Godard, O. Mac Aodha, and G. J. Brostow, “Unsupervised monocular depth estimation with left-right consistency,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [9] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao, “Deep ordinal regression network for monocular depth estimation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.
- [10] Y. Xu, X. Zhu, J. Shi, G. Zhang, H. Bao, and H. Li, “Depth completion from sparse lidar data with depth-normal constraints,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.
- [11] Y. Zhao, L. Bai, Z. Zhang, and X. Huang, “A surface geometry model for lidar depth completion,” *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4457–4464, 2021.
- [12] S. Tulyakov, D. Gehrig, S. Georgoulis, J. Erbach, M. Gehrig, Y. Li, and D. Scaramuzza, “Time lens: Event-based video frame interpolation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021, pp. 16 155–16 164.
- [13] J. Hidalgo-Carrió, D. Gehrig, and D. Scaramuzza, “Learning monocular dense depth from events,” in *2020 International Conference on 3D Vision (3DV)*, 2020, pp. 534–542.
- [14] D. Gehrig, M. Rüegg, M. Gehrig, J. Hidalgo-Carrió, and D. Scaramuzza, “Combining events and frames using recurrent asynchronous multimodal networks for monocular depth prediction,” *IEEE Robotics and Automation Letters*, vol. 6, no. 2, 2021.
- [15] H. Rebecq, D. Gehrig, and D. Scaramuzza, “Esim: an open event camera simulator,” in *Conference on Robot Learning*. PMLR, 2018, pp. 969–982.
- [16] H. Rebecq, R. Ranftl, V. Koltun, and D. Scaramuzza, “Events-to-video: Bringing modern computer vision to event cameras,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.
- [17] D. Gehrig, M. Gehrig, J. Hidalgo-Carriño, and D. Scaramuzza, “Video to events: Recycling video datasets for event cameras,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.
- [18] F. Paredes-Valles and G. C. H. E. de Croon, “Back to event basics: Self-supervised learning of image reconstruction for event cameras via photometric constancy,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021, pp. 3446–3455.
- [19] E. Coumans and Y. Bai, “Pybullet, a python module for physics simulation for games, robotics and machine learning,” <http://pybullet.org>, 2016–2021.
- [20] Y. Tassa, S. Tunyasuvunakool, A. Muldal, Y. Doron, S. Liu, S. Bohez, J. Merel, T. Erez, T. Lillicrap, and N. Heess, “dm\_control: Software and tasks for continuous control,” *arXiv preprint: 2006.12983*, 2020.
- [21] E. Todorov, T. Erez, and Y. Tassa, “MuJoCo: A physics engine for model-based control,” in *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, 2012.
- [22] R. Tedrake and the Drake Development Team, “Drake: Model-based design and verification for robotics,” 2019. [Online]. Available: <https://drake.mit.edu>
- [23] B. Wymann, E. Espié, C. Guionneau, C. Dimitrakakis, R. Coulom, and A. Sumner, “Torcs, the open racing car simulator,” *Software available at http://torcs.sourceforge.net*, vol. 4, no. 6, p. 2, 2000.
- [24] M. Müller, V. Casser, J. Lahoud, N. Smith, and B. Ghanem, “Sim4cv: A photo-realistic simulator for computer vision applications,” *International Journal of Computer Vision*, vol. 126, no. 9, pp. 902–919, 2018.
- [25] S. Shah, D. Dey, C. Lovett, and A. Kapoor, “AirSim: High-fidelity visual and physical simulation for autonomous vehicles,” in *Field and Service Robotics (FSR)*, 2017.
- [26] Y. Song, S. Naji, E. Kaufmann, A. Loquercio, and D. Scaramuzza, “Flightmare: A flexible quadrotor simulator,” *arXiv preprint arXiv:2009.00563*, 2020.
- [27] F. Xia, A. R. Zamir, Z. He, A. Sax, J. Malik, and S. Savarese, “Gibson Env: Real-World Perception for Embodied Agents,” in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [28] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra, “Habitat: A Platform for Embodied AI Research,” in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019.
- [29] S. Manivasagam, S. Wang, K. Wong, W. Zeng, M. Sazanovich, S. Tan, B. Yang, W.-C. Ma, and R. Urtasun, “Lidarsim: Realistic lidar simulation by leveraging the real world,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 167–11 176.
- [30] T.-H. Wang, S. Manivasagam, M. Liang, B. Yang, W. Zeng, and R. Urtasun, “V2VNet: Vehicle-to-Vehicle Communication for Joint Perception and Prediction,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.
- [31] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, “End to End Learning for Self-Driving Cars,” *arXiv:1604.07316*, 2016.
- [32] H. Xu, Y. Gao, F. Yu, and T. Darrell, “End-to-End Learning of Driving Models from Large-Scale Video Datasets,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [33] A. Amini, G. Rosman, S. Karaman, and D. Rus, “Variational End-to-End Navigation and Localization,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.
- [34] M. Lechner, R. Hasani, A. Amini, T. A. Henzinger, D. Rus, and R. Grosu, “Neural circuit policies enabling auditable autonomy,” *Nature Machine Intelligence*, vol. 2, no. 10, pp. 642–652, 2020.
- [35] J. Hawke, R. Shen, C. Gurau, S. Sharma, D. Reda, N. Nikолов, P. Mazur, S. Micklithwaite, N. Griffiths, A. Shah, and A. Kendall, “Urban Driving with Conditional Imitation Learning,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.
- [36] N. Rhinehart, R. McAllister, and S. Levine, “Deep imitative models for flexible inference, planning, and control,” *arXiv preprint arXiv:1810.06544*, 2018.
- [37] F. Codevilla, M. Müller, A. López, V. Koltun, and A. Dosovitskiy, “End-to-End Driving via Conditional Imitation Learning,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.
- [38] X. Pan, Y. You, Z. Wang, and C. Lu, “Virtual to real reinforcement learning for autonomous driving,” 2017.
- [39] F. Fuchs, Y. Song, E. Kaufmann, D. Scaramuzza, and P. Dürr, “Super-human performance in gran turismo sport using deep reinforcement learning,” *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4257–4264, 2021.
- [40] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl, “Learning by cheating,” in *Conference on Robot Learning*. PMLR, 2020, pp. 66–75.
- [41] F. Codevilla, A. M. Lopez, V. Koltun, and A. Dosovitskiy, “On offline evaluation of vision-based driving models,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 236–251.
- [42] A. Amini, L. Paull, T. Balch, S. Karaman, and D. Rus, “Learning Steering Bounds for Parallel Autonomous Systems,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.
- [43] A. Amini, W. Schwarting, G. Rosman, B. Araki, S. Karaman, and D. Rus, “Variational autoencoder for end-to-end control of autonomous driving with novelty detection and training de-biasing,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 568–575.
- [44] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

- [45] D. Gehrig, M. Gehrig, J. Hidalgo-Carrió, and D. Scaramuzza, “Video to events: Recycling video datasets for event cameras,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 3586–3595.
- [46] H. Jiang, D. Sun, V. Jampani, M.-H. Yang, E. Learned-Miller, and J. Kautz, “Super slomo: High quality estimation of multiple intermediate frames for video interpolation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9000–9008.
- [47] E. Mueggler, H. Rebecq, G. Gallego, T. Delbrück, and D. Scaramuzza, “The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and slam,” *The International Journal of Robotics Research*, vol. 36, no. 2, pp. 142–149, 2017.
- [48] G. Gallego, J. E. Lund, E. Mueggler, H. Rebecq, T. Delbrück, and D. Scaramuzza, “Event-based, 6-dof camera tracking from photometric depth maps,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 10, pp. 2402–2412, 2017.
- [49] Z. Liu, A. Amini, S. Zhu, S. Karaman, S. Han, and D. Rus, “Efficient and robust lidar-based end-to-end navigation,” *arXiv preprint arXiv:2105.09932*, 2021.
- [50] H. Tang, Z. Liu, S. Zhao, Y. Lin, J. Lin, H. Wang, and S. Han, “Searching Efficient 3D Architectures with Sparse Point-Voxel Convolution,” in *European Conference on Computer Vision (ECCV)*, 2020.