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

Alexander Amini\textsuperscript{1,}\textsuperscript{*}, Tsun-Hsuan Wang\textsuperscript{1,}\textsuperscript{*}, Igor Gilitschenski\textsuperscript{2}, Wilko Schwarting\textsuperscript{1}, Zhijian Liu\textsuperscript{1}, Song Han\textsuperscript{1}, Sertac Karaman\textsuperscript{1}, and Daniela Rus\textsuperscript{1}

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.

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

\textsuperscript{2} 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
Learning in simulation allows for greater algorithmic learning transferable policies for embodied AI research. With a focus on local scene synthesis for scalability and on sizing novel views. Our work follows this line of research. [4], [27]–[30] present greater photorealism by leveraging real data. In contrast, data-driven simulators [2], platforms, but still lack the photorealism necessary for direct engines rely on heavily engineered video-game rendering. 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 2.0 [4] is a data-driven simulator that synthesizes RGB images at novel viewpoints around the local trajectory of a dataset. The precollection of the 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: \((x, y, z, l)\), where \((x, y, z)\) is the position of the point in 3D Cartesian space and \(l\) 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 naïvely 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
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,

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

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” 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 (~100× faster).

Finally, we sample sparse points from our dense representation to form the novel viewpoint image. 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).
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 (> 10,000Hz) 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],

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

(2)

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

(3)

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

(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],

$$(\mathbf{p}, t_1, \ell) \text{ if } \rho (\ln(I(\mathbf{p}, t_1) - \ln(I(\mathbf{p}, t_1 - \Delta t))) \geq c_k$$

(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 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,

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

(6)

where $\Delta t_{\text{max}}$ 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,

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

(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 $T_{\text{novel}}$ 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

```plaintext
for k ← 1 to N Do
  while !buffer.full() do
    if VISTA.done() then
      VISTA.reset()
    end if
    x ← VISTA.readSensorBuffer()
    if useBranching then
      y ← privilegedController(VISTA.getState())
      revertState(VISTA, privilegedController)
    else
      y ← privilegedController(VISTA.getState())
    end if
    if VISTA.step(y) then
      if !rejectSample(x,y) then
        buffer.add(x,y)
        end if
      else
        buffer.shuffle()
        trainModel(buffer.next())
      end while
  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-23S3-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 that gradually drift off the road and cannot recover (due

| Sensor | Algo. | Mean Squared Error (1E-5) |
|--------|-------|---------------------------|
| 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].
### 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 policies exhibit greater robustness with lower crash rates.

| Sensor | Algo. | 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 |

### 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.

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 ± 3.6 (IL) 18.0 ± 1.0 (GPL) 6.5 ± 3.8 (GPL with branching). To further highlight the effectiveness of GPL policies, we conduct a robustness test by initializing the car with ±30° rotation and ±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.
[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. Delbruck, 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. Delbruck, 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:2105.09912, 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.