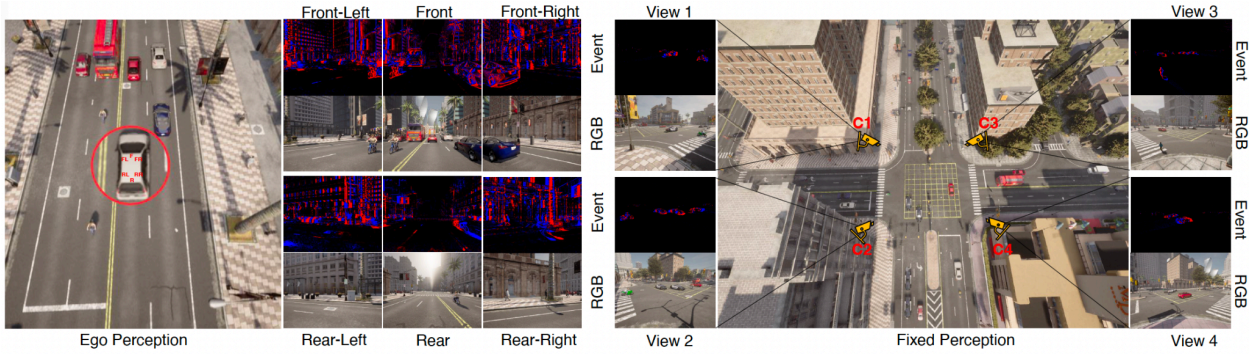
SEVD Dataset Documentation

Version: Beta (last updated on April 22, 2024)

PDF: https://arxiv.org/abs/2404.10540

Dataset:

Code: https://github.com/ eventbasedvision/SEVD



In recent years, there has been an increasing focus on neuromorphic or event-based vision due to its ability to excel under high dynamic range conditions, offer high temporal resolution, and consume less power than conventional frame-based vision sensors such as RGB cameras. The event cameras, also known as dynamic vision sensors (DVS), mimic the behavior of biological retinas by continuously sampling incoming light and generating signals only when there is a change in light intensity. This results in an event data stream represented as a sequence of ⟨x, y, p, t⟩ tuple, where (x, y) denotes pixel position, t represents time, and p indicates polarity (positive or negative contrast.

While event-based sensing represents a novel area, research efforts have been limited in recent years to fully utilize the capabilities of event-based cameras for perception tasks. Notably, researchers have predominantly used event- based cameras like DAVIS346 by iniVation and Prophesee's IMX636 / EVK 4 HD to construct automotive datasets. Additionally, researchers have employed frame-to-event simulators such as ESIM and v2e to generate synthetic event-based data. However, it only converts RGB frames of an outdoor scene from MVSEC. This highlights the significant scarcity of readily available synthetic event-based datasets in the field. To bridge this gap and leverage the potential of synthetic data to generate diverse and high-quality vision data tailored for traffic monitoring, we present SEVD – a **S**ynthetic **E**vent-based **V**ision **D**ataset designed for autonomous driving and traffic monitoring tasks. SEVD provides multi-view (360°) dataset comprising 27 hr of fixed and 31 hr of ego perception data, with over 9M bounding boxes, recorded across diverse conditions and varying

parameters. The event cameras are complemented by five different types of sensors, including RGB, depth, optical flow, semantic, instance segmentation cameras, GNSS and IMU sensors resulting in a diverse array of data. All the cameras RGB, depth, optical flow, semantic, and instance segmentation cameras are all configured to operate at a resolution of 1280 × 960 px. Data collection sites are listed below.

Map	Description
Town10HD_Opt	A bustling urban landscape boasting gleaming skyscrapers and industrial edifices.
Town03	A suburban area featuring a network of roads including a roundabout and sloping intersections.
Town04	A highway-centric town featuring underpasses, overpasses, and circular slip roads within the urban environment.
Town05	Suburban surroundings characterized by 3-way intersections, elevated highways, and extensive multi lane roads.
Town07	A rural community nestled amidst green fields, barns, grain silos, and windmills.

For the coordinates of fixed perception cameras at each map, refer to the GitHub code at:

SEVD offers a diverse range of recordings featuring various combinations of scenes, weather, and lighting conditions. The simulation operates in synchronous mode with a time-step of 0.1 seconds, corresponding to 10 frames per second (fps). The dataset is organized using the following folder structure.

```
## fixed perception

id
├ fixed-x1
│    ├── dvs_camera
│    ├── rgb_camera
│    └── optical
├ annotations
│    ├── dvs_camera
│    └── rgb_camera
├ xytp
│    └── fixed-x1
│    └── fixed-x2 ...
├ metadata.json // Metadata includes town, weather, duration, participant count


## ego perception

id
├ dvs_camera
│  - dvs_camera-back
│  - dvs_camera-back-right
│  - dvs_camera-back-left
│  - dvs_camera-front
│  - dvs_camera-front-left
│  - dvs_camera-front-right
├ similarly for rgb_camera, optical_flow, semantic_segmentation,
instance_segmentation, xytp
├ gnss
│  - gnss.txt
├ imu
│  - imu.txt
├ metadata.json
```

DVS raw streams are provided in `.npz` format as a sequence of ⟨x, y, t, p⟩ tuple. Additionally, we also provide code for custom data generation at https://github.com/eventbasedvision/SEVD.

Following the simulation phase, we employ CARLA's bounding box API to generate annotations in both 2D and 3D bounding box formats, such as COCO, Pascal VOC, and KITTI. In each of the annotations, both class_id and object_id are provided for every actor.