Conversion Requirements for IDD3D   
Detailed table of comparison:

|  |  |  |  |
| --- | --- | --- | --- |
| x | Nuscenes | IDD3D | Changes required |
| Sensors/ Primary modalities | 6 cameras, 5 radars, 1 lidar, GPS, IMU all with 360-degree FOV | 6 cameras, 1 LiDAR, GPS, raw sensor files (in ros-bag format) |  |
| Sensor Specs | * 6 Cameras – 12Hz capture frequence, 1600x900, JPEG compressed * LiDAR – Velodyne HDL-32E, 32 channel, 20 Hz, .pcd.bin, stores x, y, z, intensity, and ring values per point * 5 Radars – 13hz * GPS/IMU - 1k Hz update rate | * 6 RGB cameras - 10Hz capture, 1440x1080, Png format(2048 x 1536 in paper) * 1 LiDAR – 64 channel(v), 1024 channel(h), 10Hz capture, pcd * GPS- BU-353-S4 sensor ,1Hz (mentioned) | According to pipeline requirements format conversion |
| Scenes | 1000 scenes of 20s duration | 150 sequences ,100 frames at 10fps, each of 10s. |  |
| Annotations | 3D bounding boxes   * obj instances and tracking IDs at 2Hz * 23 object classes * Using pointpillar * 40k Ann frames with 1.4M 3D bounding boxes   Map layers – total 11 map layers like drivable areas, lanes, etc. | 3D bounding boxes   * 9DoF * Instance IDs for tracking (maybe at 10hz) * 17 classes (10 primary + 7 misc) * Annotations provided per lidar frame/per sequence * 15.5k (train-val-test) LiDAR frames with 223k 3D bounding boxes | Will have to make a converter template |
| Format | Images(jpg), lidar sweeps (.pcd.bin), radar (pcd), JSON metadata | Images (Png), lidar(pcd), annotation Json, Original data in rosbag format, |  |
| Timestamps and sync | data alignment between the lidar and the cameras – timestamps provided in unix ns | (Camera, lidar and labels share same numeric id in filenames and annot\_data.json groups frames together ) | Time stamps generated, lidar data converted as per format needed |
| Calibration and extrinsic | Provided calibrated\_sensor.json +ego\_pose.json | Calib for camera sensor file present but empty!! Also, no file for lidar | Done except egopose |
| Maps | semantic maps, original rasterized map(only roads and sidewalks) + vectorized map expansion(11 classes | Not present | Generate HD map |

IDD3D data format

IDD3D dataset  
└── train\_val   
 └── 20220118103308\_seq\_10   
 ├─ 20220118103308\_seq\_10

├ ── annot\_data.json   
 ├ ── calib   
 ├ ── camera   
 ├ ── label   
 └── lidar   
 ├ ── annot\_data.json   
 ├ ── calib   
 ├ ── camera   
 ├ ── label   
 └── lidar

One sequence-3.1gb

TiAND dataset

|  |  |  |  |
| --- | --- | --- | --- |
| x | Nuscenes | TiAND | Changes required |
| Sensors/ Primary modalities | 6 cameras, 5 radars, 1 lidar, GPS, IMU all with 360-degree FOV | 4 cameras, 1 lidar, 6 radars (5 short-range + 1 long-range), GNSS and IMU |  |
| Sensor Specs | * 6 Cameras – 12Hz capture frequence, 1600x900, JPEG compressed * LiDAR – Velodyne HDL-32E, 32 channel, 20 Hz, pcd * 5 Radars – 13hz * GPS/IMU - 1k Hz update rate | * 4cameras – 1280x720, JPG format * 1 LiDAR – 10fps, pcd , 360degree * 6 radars – 5 short range + 1 long range radar * GNSS | According to pipeline requirements format conversion |
| Scenes | 1000 interesting scenes of 20s duration | 150 scenes, each of 2-4 min scenes (= mentioned in paper – but website downloaded dataset only of 15Gb) |  |
| Annotations | 3D bounding boxes   * obj instances and tracking IDs at 2Hz * 23 object classes * Using pointpillar * 40k Ann frames with 1.4M 3D bounding boxes   Map layers – total 11 map layers like drivable areas, lanes, etc. | No annotation files in downloaded dataset  (Annotations in YOLO format across 24 relevant object categories mentioned in paper) | Will have to make a converter template |
| Format | Images(jpg), lidar sweeps (pcd.bin), radar (pcd), JSON metadata | Images (jpg), lidar (pcd), radar (csv), gnss(.csv), calib(.npy), sync(.csv) |  |
| Timestamps and sync | data alignment between the lidar and the cameras – timestamps provided in unix ns | Synchronisation between lidar-camera and radar-camera present. |  |
| Calibration and extrinsic | Provided calibrated\_sensor.json +ego\_pose.json | Calibration files of lidar and radar present. |  |
| Maps | semantic maps, original rasterized map(only roads and sidewalks) + vectorized map expansion(11 classes | No |  |

For IDD3d and TiAND:  
  
1) Map -   
  
 NuScenes map format –

* Top down, image-based representation of semantic layers with 11 semantic classes(lane dividers, lane boundaries, crosswalks, sidewalks, stop lines, drivable area, etc).
* It also provides baseline routes - the idealized path an AV should take, assuming there are no obstacles.
* UniAD focuses on drivable areas, lane boundaries, and pedestrian crossings.

To generate maps in idd3d and tiand-

Both idd3d and tiand supply lidar point clouds, so we can derive occupancy from point projections and camera images could be fed to a segmentation model to label lanes/road surface.  
  
some methods:

1. HD map inference:  
   HDMapNet -An Online HD Map Construction and Evaluation Framework  
   HDMapNet encodes image features from surrounding cameras and/or point clouds from LiDAR and predicts vectorized map elements in the bird's-eye view.  
     
   Similarly VectorMapNet – predicts vectorized map elements for lane/dividers/crossings.  
   We could train or run these or just note that similar architectures could provide lane polylines.
2. OpenStreetMap(OSM)  
   With GNSS of tiand, coarse road network data from OSM can provide lane and topology by aligning each trajectory to OSM.
3. Full HD map  
   aggregate lidar into a global point cloud, vectorizing lanes, generate raster layers, export to JSON

IDD3D – To generate maps, we can

HD mapping tools like HDMapNet   
  
or

* first perform per frame semantic segmentation on each camera image (with nuscenes or cityscapes trained models) to label lane marking, etc.
* These labels can be projected to BEV using camera intrinsics/extrinsic and Lidar depth (taking ground points to show drivable area by e.g. RANSAC).
* Can vectorize it further if needed by extracting lane boundary or crosswalk lines to form vector map.

TiAND TiHAN dataset –

Same as IDD3D or

* We can use its GPS data and OSM for local road network.
* Then we can integrate camera images for lane level details and Lidar to refine drivable area.

(HD mapping tools also can be used for all)

2) CAN – vehicle's low level odometry controls  
  
NuScenes CAN – vehicle speed, steer angle feedback, wheel speeds, acceleration, etc. It is timestamped and aligned to sensor frames.

I) Vision based prediction -   
VLM auto gives textual output.   
Can train VLM to give numerical output for CAN (like in bench2advlm) .   
  
II) Tiand dataset has GNSS/IMU which outputs latitude, longitude, altitude, heading and north/east velocity components, etc. So, some CAN variables can be estimated with the help of ego pose information. Speed form north and east speed, acc – differentiate velocity data, steering angle – more complex formula – bicycle model.  
Maybe we’ll need to smoothen raw gnss data to reduce noise.

III) For IDD3D – not sure of gps data availability and ego pose. (odometry and gps mentioned in supplementary paper but not in the main paper)

IV) Simulation based generation – Realistic simulator (CARLA, AirSim) can be set up with Indian road scenes to generate CAN data.

3) Conversion of IDDM logs into JSON format containing sample, sensor calibration, ego\_pose, annotations. Token generation or consistent IDs for all entries (as in nuScenes , each piece of data is refereed by unique token.

4) Annotations – re-format annotations

5) Calibration