Supplementary Materials To: FusionPortable: A Multi-Sensor Campus-Scene Dataset for Evaluation of Localization and Mapping Accuracy on Diverse Platforms

This supplementary material mainly describes details of our dataset and evaluation results, including calibration, example data of the dataset, and SLAM results of some SOTA SLAM algorithms.

I. Calibration

A. Stereo Camera Calibration

We use the Matlab calibration toolbox[1] to calibrate cameras' intrinsics and extrinsics using a checkerboard ($3cm$ pattern). We move the sensor suite before the checkerboard to collect a sequence of images. To avoid motion blur and ensure sufficient constraints, the motion is slow and contains rich transformation. We use the pinhole camera and radial-tangential distortion model according to the lens used on cameras. Matlab provides user-friendly GUI for users to manually select high-quality data by removing images with high-reprojection error, large roll angles, and wrong pattern detection. Fig. 1 shows an example of this GUI. After getting calibrated intrinsics and extrinsics, we can verify them via. Matlab functions related to depth estimation. A simple tutorial is documented[2], where stereo image is first rectified (`rectifyStereoImages`), the disparity map is then computed (`disparitySGM`), and the 3D scene is finally reconstructed (`reconstructScene`). Fig. 2(a) shows an example of rectified stereo images. Fig. 2(b) and Fig. 2(c) show the reconstruction results of stereo frame camera and stereo event camera respectively.

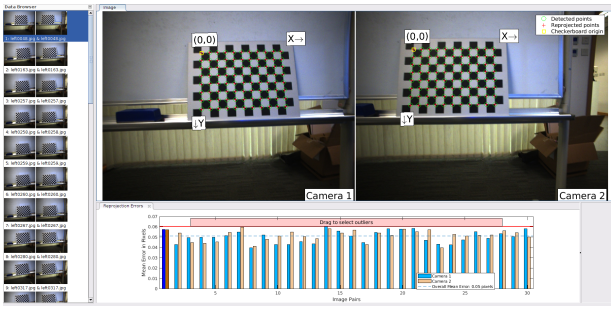
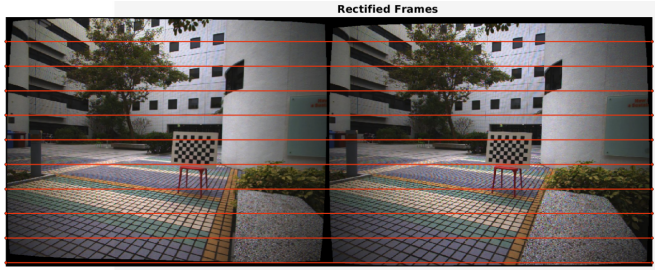
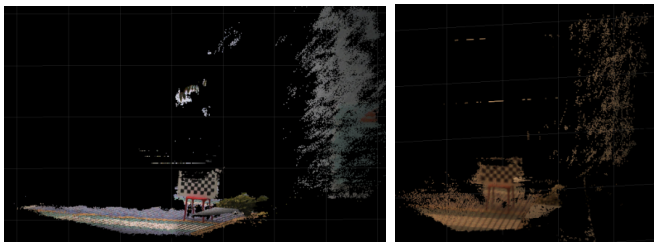


Fig. 1. Stereo camera calibration via. Matlab.



(a)



(b) (c)

Fig. 2. Immediate results of camera calibration: (a) rectified stereo frame images, (b) reconstruction by stereo frame camera, and (c) reconstruction by stereo event camera.

B. Camera-LiDAR Extrinsic Calibration

A checkerboard ($6.8cm$ pattern) is used to provide distinct geometric features for calibration. After getting extrinsics, we can project LiDAR points onto the image plane for verification. Fig. 3(a) and Fig. 3(b) visualize the LiDAR projection results onto frame cameras and event cameras respectively.

C. Hand-Eye Calibrtion

II. Dataset Description

A. Ground-truth Maps

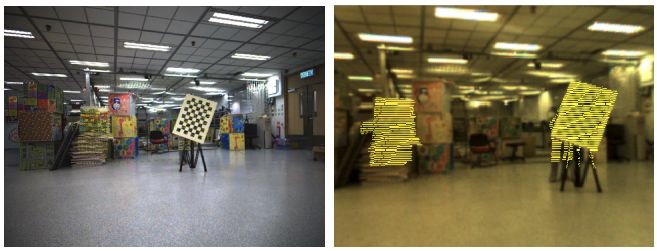
Fig. 4(a) and Fig. 4(b) shows a part of ground-truth maps captured in the corridor and building scenarios. The resolution is $1cm^3$.

B. Drift Caculation

For sequences where groundtruth is unavailable, we calculate the drift for algorithm evaluation. We place a checkerboard at the start frame and end frame of the sequence. By exploiting the patterns from the checkerboard, we can calculate the relative rotation and translation ($\boldsymbol{R}_1$ and $\boldsymbol{t}_1$) between these two frames. For SLAM algorithms, they can also estimate the rotation and translation ($\boldsymbol{R}_2$ and $\boldsymbol{t}_2$). Denot-

[1] `https://ww2.mathworks.cn/help/vision/camera-calibration.html`

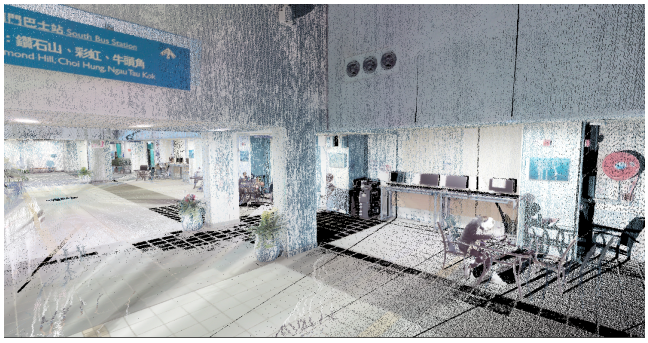
[2] `https://ww2.mathworks.cn/help/vision/ug/depth-estimation-from-stereo-video.html`



(a) (b)

Fig. 3. LiDAR points are projected onto the (a) frame camera and (b) event camera. The resolution of these cameras are 768×1024 and 260×346 respectively.

(a) Classroom in the corridor



(b) Building

Fig. 4. Ground-truth point cloud in color of the classroom in the corridor and building scenario. Point cloud data was recorded by the Leica BLK360 laser scanner. They are used to generate trajectory groundtruth and evaluate algorithms' reconstruction accuracy.

ing $d\boldsymbol{R} = \boldsymbol{R}_1^\top \boldsymbol{R}_2$ and $d\boldsymbol{t} = -(\boldsymbol{R}_1^\top \boldsymbol{t}_2) + \boldsymbol{t}_1$, we can compute the rotational and translation drift as $\theta = |\arccos(\frac{\mathrm{tr}(d\boldsymbol{R})-1}{2})|$ and $t = ||d\boldsymbol{t}||$.

C. Format of Calibration Parameters

Sensor calibration parameters are stored in `yaml` files. Taking the right frame camera as an example, Fig. 5 show the format of calibration files. Camera intrinsics including the intrinsic matrix, distortion matrix, rectification matrix, and projection matrix. Camera extrinsics are represented using the Hamilton quaternion (i.e., $[w, x, y, z]^\top$) for rotation and 3×1 vector for translation. Here, `quaternion_stereo` and `translation_stereo` indicates the extrinsics and the sensor itself to the left frame camera, and `quaternion_sensor_bodyimu` and

```yaml
dataset_name: calib_20220209
image_width: 1024
image_height: 768
camera_name: stereo_right_Flir_BFSU3
camera_matrix: !!opencv-matrix
  rows: 3
  cols: 3
  dt: f
  data: [ 6.04964966e+02, 0., 5.17844666e+02,
          0., 6.04625610e+02, 3.89209320e+02,
          0., 0., 1. ]
distortion_model: plumb_bob
distortion_coefficients: !!opencv-matrix
  rows: 1
  cols: 4
  dt: f
  data: [ -9.58003029e-02, 8.74120295e-02, 2.08094658e-04, -1.08567670e-04 ]
rectification_matrix: !!opencv-matrix
  rows: 3
  cols: 3
  dt: f
  data: [ 9.99987543e-01, 4.83056623e-03, 1.24577642e-03,
          -4.83735651e-03, 9.99973118e-01, 5.50653692e-03,
          -1.21914328e-03, -5.51249459e-03, 9.99984086e-01 ]
projection_matrix: !!opencv-matrix
  rows: 3
  cols: 4
  dt: f
  data: [ 6.04799866e+02, 0., 5.14155396e+02, -9.65739136e+01,
          0., 6.04799866e+02, 3.91754669e+02, 0.,
          0., 0., 1., 0. ]
quaternion_stereo: !!opencv-matrix
  rows: 1
  cols: 4
  dt: f
  data: [ 9.99922812e-01, 5.50936209e-03, 3.47419176e-03, 1.05790943e-02 ]
translation_stereo: !!opencv-matrix
  rows: 1
  cols: 3
  dt: f
  data: [ -1.59677148e-01, -7.71340623e-04, -1.98924507e-04 ]
quaternion_sensor_bodyimu: !!opencv-matrix
  rows: 1
  cols: 4
  dt: f
  data: [0.495420, 0.501199, -0.503827, 0.499516]
translation_sensor_bodyimu: !!opencv-matrix
  rows: 1
  cols: 3
  dt: f
  data: [-0.093388, -0.017886, -0.078768]
timeshift_sensor_bodyimu: 0.035039150765962465
```

Fig. 5. Calibration parameters of the right frame camera.

`translation_sensor_bodyimu` indicates the extrinsics and the sensor itself to the body IMU (STIM300). Benefiting the Kalibr package [1], we can also obtain the time offset: `timeshift_sensor_bodyimu`. But we cannot guarantee that this value is exact for specific usage.

III. Experiment

A. Localization Results

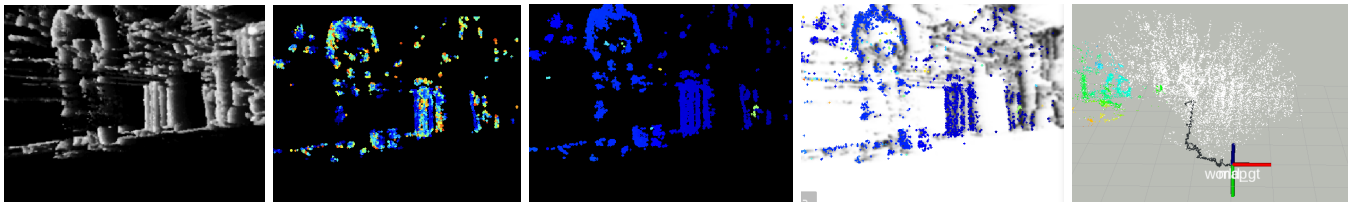
1) Results of ESVO: ESVO was tested with this dataset. Here, we partially show ESVO's tracking and mapping results on the canteen_night, MCR_slow_00, and campus_road_day sequences.

2) Results of Other SLAM Algorithms: We provide more results regarding different data collection platforms:

1) Handheld: Fig. 9 shows estimated trajectories against the ground truth on cateen_night, canteen_day, garden_night, corridor_day, building_day, and MCR_normal sequences respectively. Fig. 10 shows the relative pose error of algorithms on the MCR_normal sequence.
2) Quadruped Robot: Fig. 11 shows estimated trajectories against the ground truth on the MCR_slow_00, MCR_normal_00, MCR_fast_01 sequences respectively.
3) Apollo: Fig. 13 shows the relative pose error of algorithms on the MCR_fast_00 sequence.

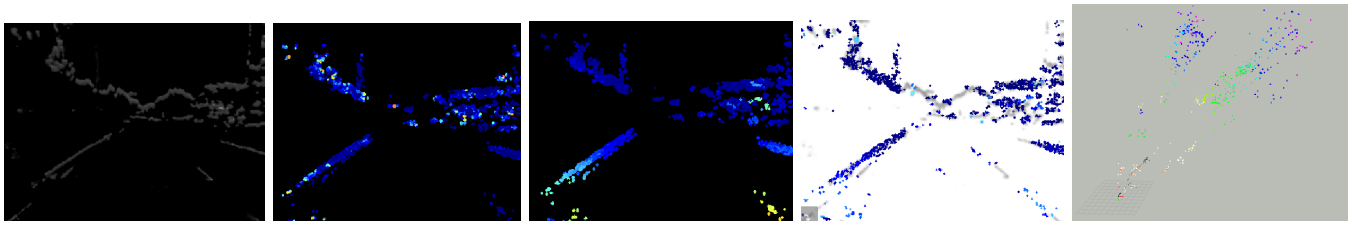
(a) (b) (c) (d) (e)

Fig. 6. Immediate results of ESVO on the canteen_night (handheld) sequence: the (a) time surface map, (b) cost map, (c) inverse depth map, (d) reprojection map on the inverse time surface, and (e) localization and mapping (colored: local map, white: global map) results.



(a) (b) (c) (d) (e)

Fig. 7. Immediate results of ESVO on the MCR_slow_00 (quadrupedal robot) sequence: the (a) time surface map, (b) cost map, (c) inverse depth map, (d) reprojection map on the inverse time surface, and (e) localization and mapping (colored: local map, white: global map) results.



(a) (b) (c) (d) (e)

Fig. 8. Immediate results of ESVO on the campus_road_day (apollo) sequence: the (a) time surface map, (b) cost map, (c) inverse depth map, (d) reprojection map on the inverse time surface, and (e) localization and mapping (colored: local map, white: global map) results.

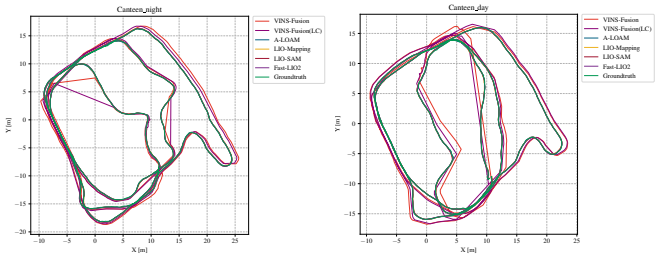
B. Mapping Results and Evaluation

In addition to the manuscript, we evaluate the algorithms' mapping results on the canteen_day, garden_night, building_day, escalator_day, and MCR_normal sequences in Fig. 14, 15, 16, 17, and 18.

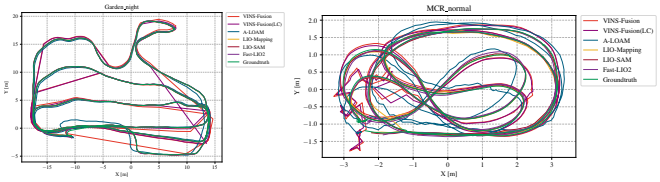
IV. Known Issues

References

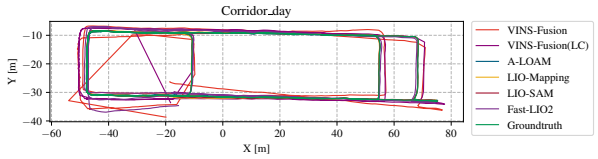
[1] P. Furgale, J. Rehder, and R. Siegwart, "Unified temporal and spatial calibration for multi-sensor systems," in 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2013, pp. 1280–1286.



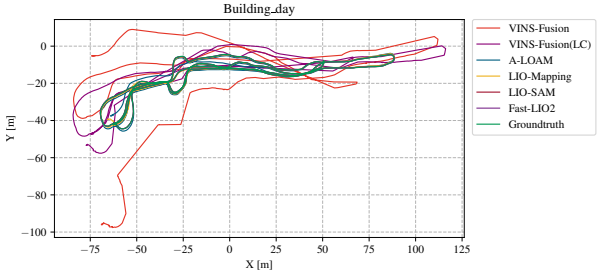
(a) canteen_night (b) canteen_day



(c) garden_night (d) MCR_normal



(e) corridor_day



(f) buliding_day

Fig. 9. Trajectory on the (a) canteen_night, (b) canteen_day, (c) garden_night, (d) MCR_normal, (e) cooridor_day, and (f) building_day sequences.

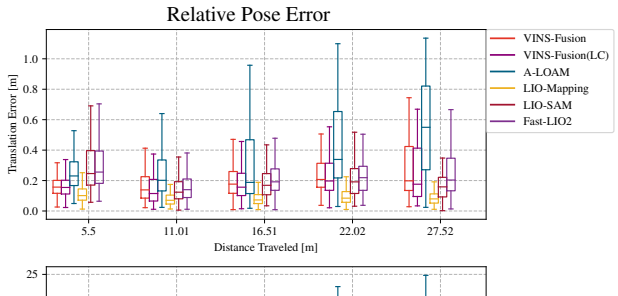
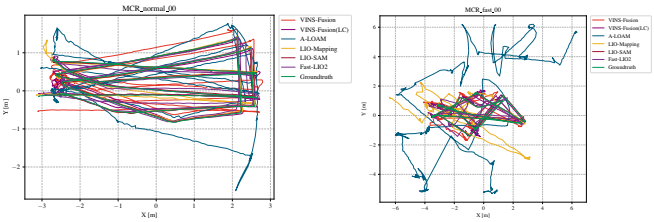
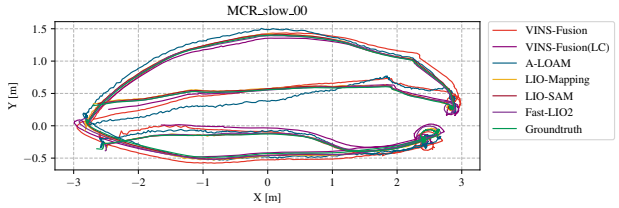


Fig. 10. Relative pose error of algorithms on the MCR_normal sequence.



(a) MCR_normal_00 (b) MCR_fast_00



(c) MCR_slow_00

Fig. 11. Trajectory on the (a) MCR_slow_00 (b) MCR_normal_00, and(c) MCR_fast_00 sequence.

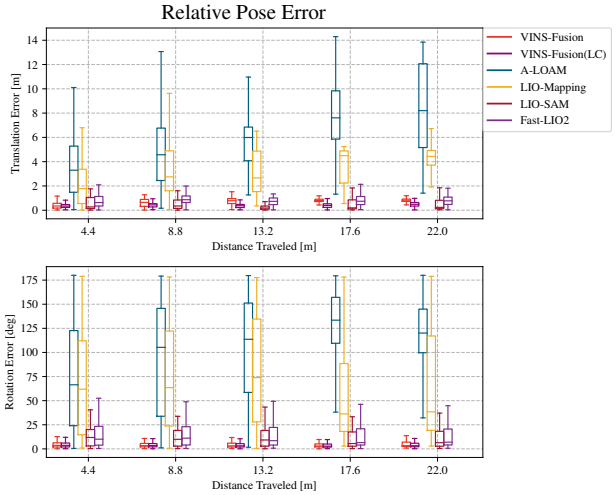


Fig. 12. Relative pose error of algorithms on the MCR_fast_00 sequence.

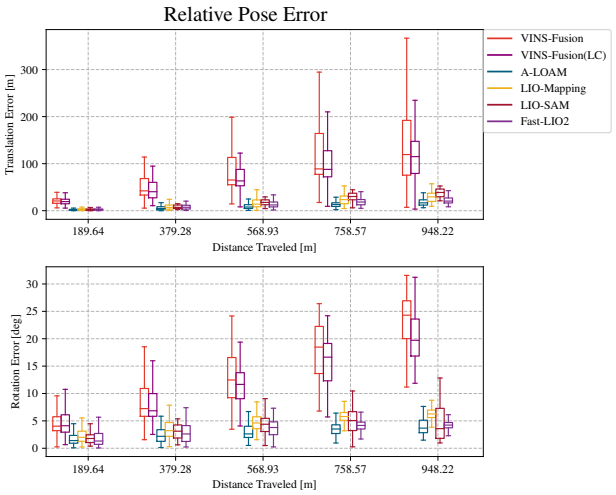
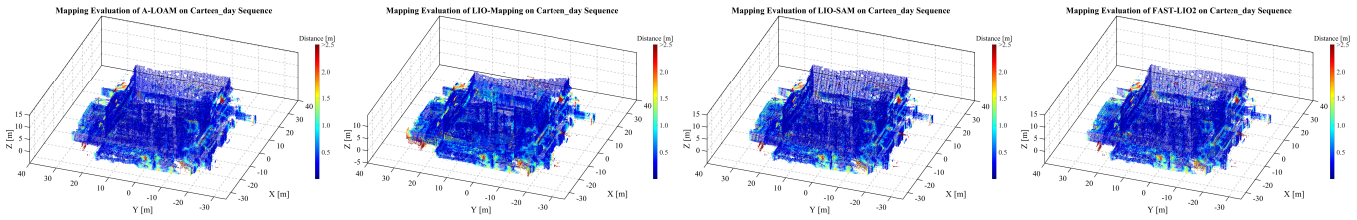
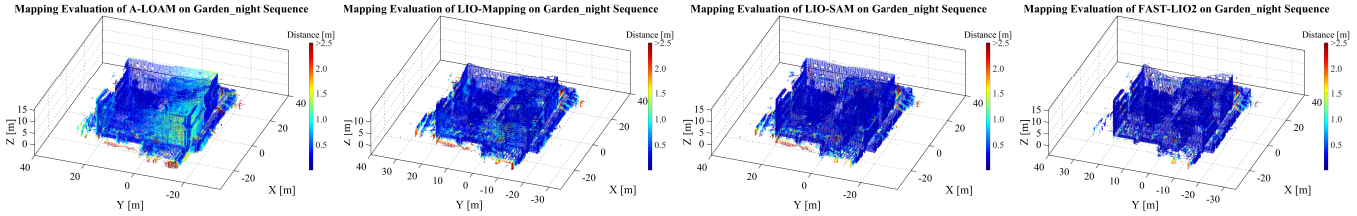


Fig. 13. Relative pose error of algorithms on the campus_road_day.



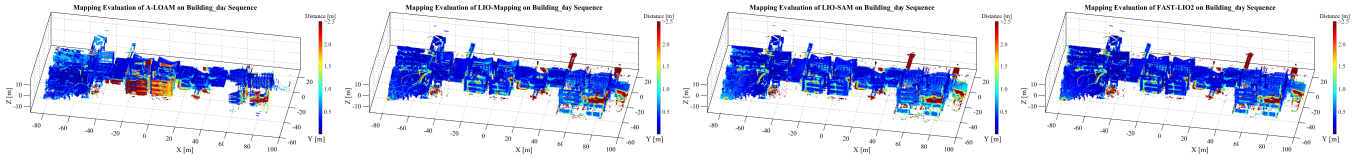
(a) A-LOAM (b) LIO-Mapping (c) LIO-SAM (d) FAST-LIO2

Fig. 14. Evaluation of (a) A-LOAM's, (b) LIO-Mapping's, (c) LIO-SAM's, and (d) FAST-LIO2's mapping accuracy on the canteen_day sequence.



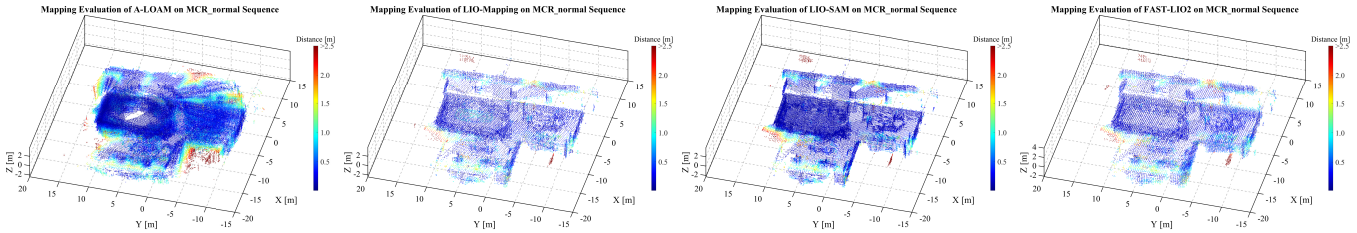
(a) A-LOAM (b) LIO-Mapping (c) LIO-SAM (d) FAST-LIO2

Fig. 15. Evaluation of (a) A-LOAM's, (b) LIO-Mapping's, (c) LIO-SAM's, and (d) FAST-LIO2's mapping accuracy on the garden_night sequence.



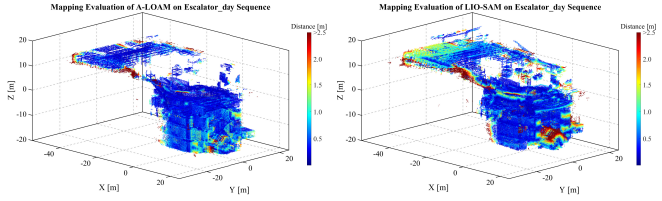
(a) A-LOAM (b) LIO-Mapping (c) LIO-SAM (d) FAST-LIO2

Fig. 16. Evaluation of (a) A-LOAM's, (b) LIO-Mapping's, (c) LIO-SAM's, and (d) FAST-LIO2's mapping accuracy on the building_day sequence.



(a) A-LOAM (b) LIO-Mapping (c) LIO-SAM (d) FAST-LIO2

Fig. 17. Evaluation of (a) A-LOAM's, (b) LIO-Mapping's, (c) LIO-SAM's, and (d) FAST-LIO2's mapping accuracy on the MCR_normal sequence.



(a) A-LOAM (b) LIO-SAM

Fig. 18. Evaluation of (a) A-LOAM's and (b) LIO-SAM's mapping accuracy on the escalator_day sequence.