Dataset: Consistent Decentralized Cooperative Localization for Autonomous Vehicles using LiDAR, GNSS and HD maps

Elwan Héry, Stéphane Bonnet, Anthony Welte, Philippe Xu and Philippe Bonnifait
Université de Technologie de Compiègne
CNRS, Heudiasyc UMR 7253
CS 60 319, 60 203 Compiègne cedex, France
www.hds.utc.fr
elwan.hery@hds.utc.fr

Dataset recording: 27 July 2018 17:53
Documentation version: 20 December 2020

This dataset is published with the journal paper:
Héry E, Xu Ph, Bonnifait Ph. Consistent Decentralized Cooperative Localization for Autonomous Vehicles using LiDAR, GNSS and HD maps. J Field Robotics. 2020;0-0. https://doi.org/10.1002/rob.22004

1 Experimental platform

This dataset was recorded for cooperative localization with two Renault Zoés.

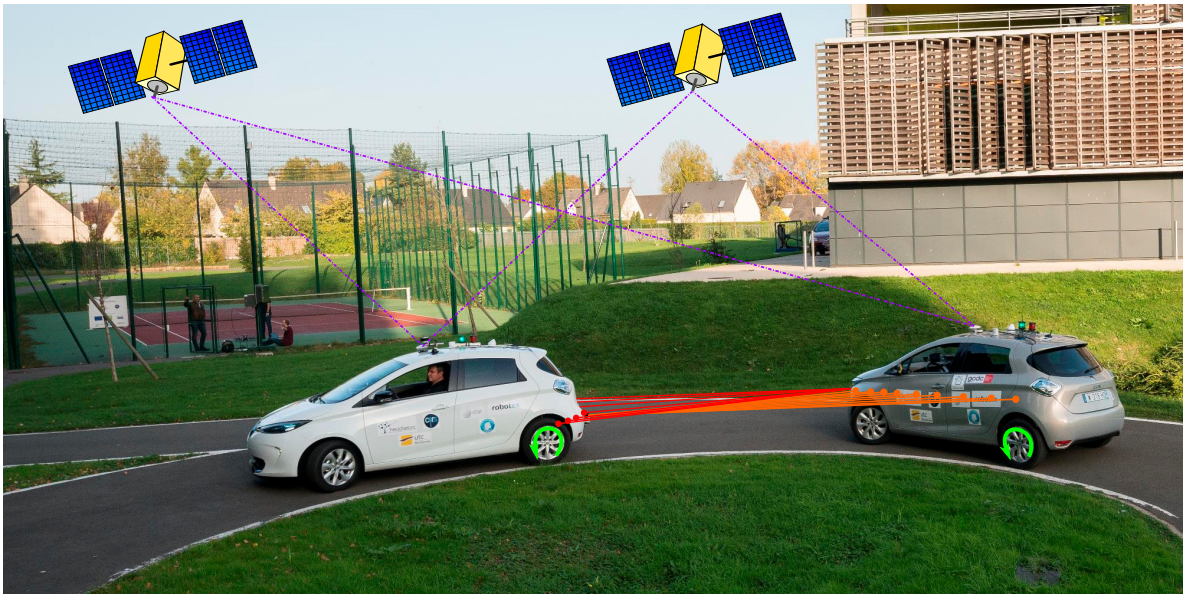


Figure 1: Experimental platform of the Heudiasyc UMR UTC/CNRS 7253 lab with two vehicles equipped with low cost GNSS receivers (in purple), dead reckoning sensors (in green) and LiDARs (in red and orange).

*https://creativecommons.org/licenses/by-nc-sa/4.0/

Figure 1 shows a platooning of the two Renault Zoés on the UTC Seville test track in Compiègne, France. Each vehicle is equipped with a low-cost mono-frequency GNSS receiver Ublox M8T EVK for global pose estimation and an RTK GNSS/INS receiver Novatel SPAN CPT for the ground truth (in purple). The dead reckoning information is estimated by the vehicle and sent through the CAN bus (in green). Finally, the following vehicle is able to perceive the leading vehicle with a 110° field of view SICK LDMRS LiDAR (in red) and the leader can perceive the follower with a SICK LMS 151 LiDAR.

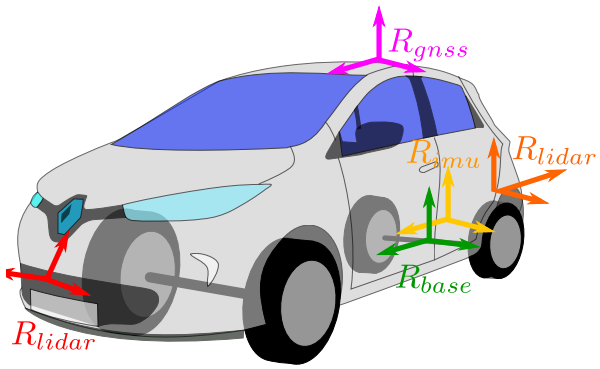


Figure 2: Frames of the vehicle (in green), the ground truth (in yellow), the GNSS receiver antenna (in purple) and the front and back LiDAR (in red and orange).

The global poses are given on an ENU (East, North, Up) frame centered at the following position:

- latitude = 49.399175564°
- longitude = 2.798755219°
- altitude = 73.5217 m (relative to the reference WGS84 ellipsoid)

Each sensor uses a different frame:

- the position of the ground truth in the frame of the base is (-0.093 0.0 0.202)
- the position of the GNSS antenna in the frame of the base is (1.388 0.0 1.302)
- the position of the front LiDAR in the frame of the base is (3.284 0.0 0.075) with 0.4° pitch rotation (as illustrated in figure 2).
- the position of the back LiDAR in the frame of the base is (-0.807 0.0 0.160) with 180° yaw rotation.

2 Scenarios

2.1 Platooning Seville

Figure 3: Test track with the nine lapses of the leader (in blue) and the follower (in green) vehicles. *(Leaflet | Tiles © Esri - Source: Esri, i-cubed, USDA, USGS, AEX, GeoEye, Getmapping, Aerogrid, IGN, IGP, UPR-EGP, and the GIS User Community).*

The two Renault Zoés are driving on a platooning in the Seville test track composed of two roundabouts and a straight lane. Nine lapses were done on the 10 min drive as illustrated in figure 3.

3 Calibration

3.1 GNSS

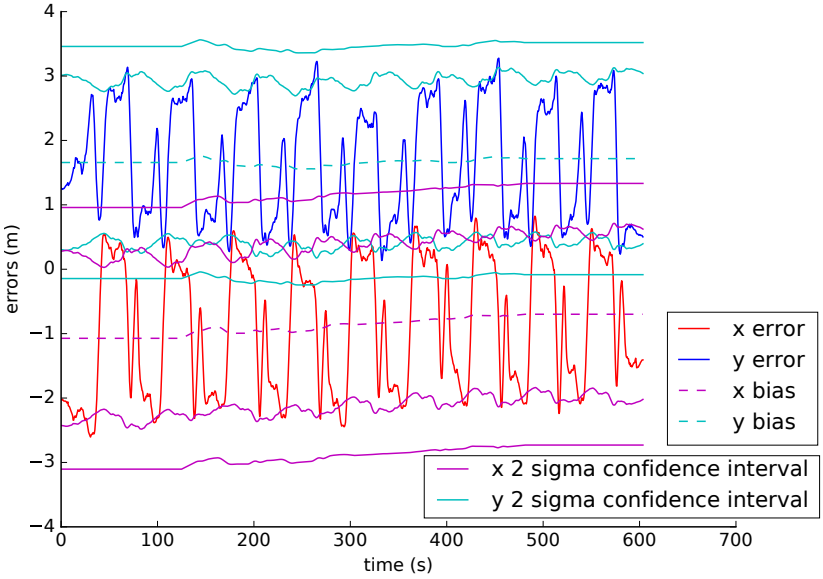


Figure 4: Errors of the leader Ublox 8 GNSS receiver with the biases estimation and confidence intervals.

Figure 4 show the GNSS biases estimated in post-processing with a mean filter. The 2σ confidence intervals are computed from the "horizontal accuracy" field provided by the Ublox 8 (for the smaller intervals) and by standard deviations computed with these errors (for the Bigger intervals). The estimated standard deviations are for the follower:

$\sigma_x = 1.055\ m$
$\sigma_y = 0.885\ m$
$\sigma_\theta = 0.0398\ rad$
and for the leader:
$\sigma_x = 1.016\ m$
$\sigma_y = 0.901\ m$
$\sigma_\theta = 0.0362\ rad$

3.2 Speed and yaw rate

Sensor calibration reduces errors due to incorrect parameters. For example, an uncalibrated gyrometer can return a biased yaw rate measurement.

In the case of dead reckoning, the current heading depends only on the initial heading and the yaw rate. On the figure 5, we observe that the heading error has a slope which is due to a bias on the yaw rate given by the ESP gyrometer. This bias is estimated by averaging the signed error between the yaw rate coming from the CAN bus and the yaw rate estimated by the ground truth. In one test, a bias of 0.004075 rad/s was observed for the following vehicle and a bias of 0.003463 rad/s for the leading vehicle. Removing this bias eliminates the observed slope (Fig. 5).

Once the heading error has been corrected, the longitudinal error can then be corrected. If we consider only straight lines, when the yaw rate is close to 0, we can see that the speed error is constantly greater than 0 (in red on the figure 6a). As the perimeters of the wheels can change according to the condition of the tires, it is important to be able to estimate them regularly to avoid incorrect speeds for each wheel. In practice, this phenomenon can be considered as a scaling factor on speed. After calibration, we found a factor of 1.01 for both vehicles. Thus the average longitudinal error when the vehicle is moving in a straight line is considerably reduced (in blue in the figure 6b).

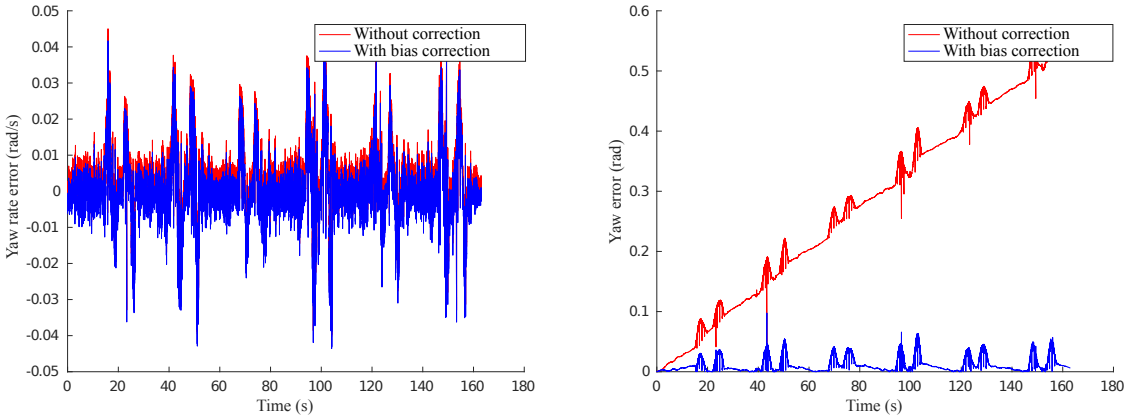
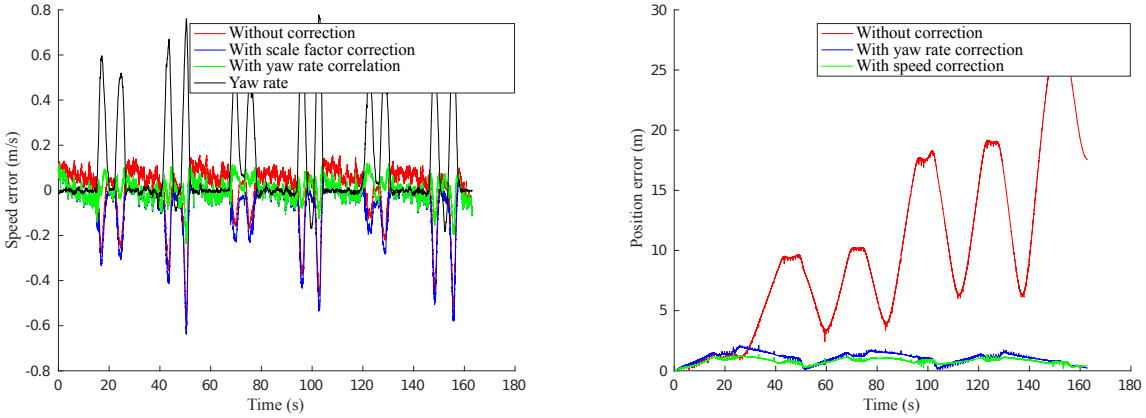


Figure 5: Yaw rate correction and influence on the heading of the vehicle.



(a) Speed error.

(b) Position error.

Figure 6: Speed errors of the leader Vehicle with and without corrections and influence on position errors.

Table 1: Position error $\|\bar{\epsilon_p}\|$, speed error $|\bar{\epsilon_v}|$, heading error $|\bar{\epsilon_\theta}|$ and yaw rate error $|\bar{\epsilon_\omega}|$ without correction and with correction of the gyrometer bias (0.003325 rad/s), by adding a scale factor (1.0157) and then using a lever arm (0.5284 m). These corrections are cumulated to reduce the errors as much as possible.

	Without correction	Gyro bias	Scale factor	lever arm
$\|\bar{\epsilon_\omega}|$ (rad/s)	0.006768	0.006202	0.006202	0.006202
$\|\bar{\epsilon_\theta}|$ (rad)	0.2819	0.0117	0.0117	0.0117
$\|\bar{\epsilon_v}|$ (m/s)	0.088207	0.088207	0.083360	0.037656
$\|\bar{\epsilon_p}\|$ (m)	10.0807	1.0799	1.2150	0.8014

Once this error has been corrected, we can see that the remaining error seems to be correlated to the yaw rate of the vehicle (in blue on the figure 6a). Indeed, the tires of the vehicle are deformed in turns. As the ESP longitudinal speed estimation is based only on a kinematic model and not on a dynamic one, the resulting error is retained and is not given at the center of the rear axle and the yaw rate intervenes. By adding the absolute value of the yaw rate multiplied by a lever arm, the longitudinal speed can be corrected (in green in the figure 6a).

The table 1 shows the errors of yaw rate, heading, longitudinal speed and position. When the gyrometer bias is corrected, the yaw rate error is reduced and the heading and position errors are also reduced. The scaling factor applied to the speed then reduces its error. Finally, the addition of the lever arm makes the speed and position errors smaller.

The final corrections are:

$$\omega = \omega + \omega_0 \tag{1}$$

$$v = k_v \cdot v - l \cdot |\omega| \tag{2}$$

With for the follower:
$\omega_0 = 0.003264\, rad/s$
$k_v = 1.0124$
$l = 0.4847\, m$
and for the follower:
$\omega_0 = 0.003325\, rad/s$
$k_v = 1.0157$
$l = 0.5284\, m$

After this correction the standard deviation of the yaw rate and the speed is estimated for the follower:
$\sigma_v = 0.103\, m/s$
$\sigma_\omega = 0.0447\, rad/s$
and for the leader:
$\sigma_v = 0.106\, m/s$
$\sigma_\omega = 0.0435\, rad/s$

4 Post processing

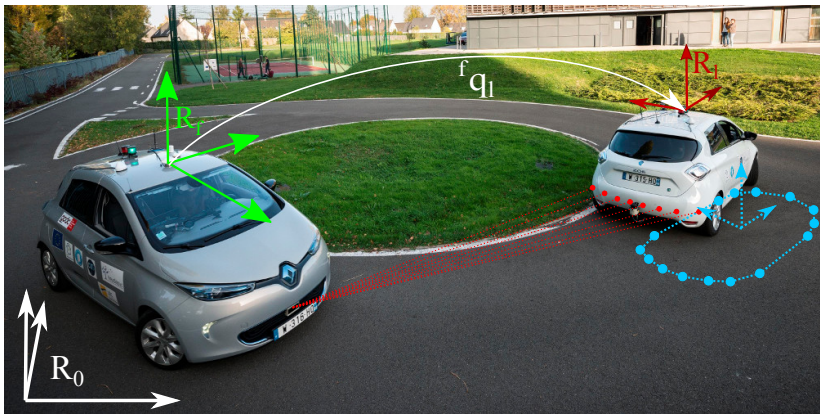


Figure 7: Relative pose between the leader and the follower vehicles estimated from the LiDAR points (in red) and the polygonal model (in blue).

The relative pose between the leader and the follower vehicles is estimated with a point to line iterative closest point (PLICP) (Figure 7). The SICK LDMRS LiDAR points (in red) are matched with a polygonal model (in blue). The relative pose is estimated by minimizing the mean distance between the LiDAR points and the model. The model is initialized using the Novatel SPAN CPT GNSS receivers to compute the initial relative pose. This algorithm is presented in:

E. Héry, Ph. Xu, and Ph. Bonnifait. LiDAR based relative pose and covariance estimation for communicating vehicles exchanging a polygonal model of their shape. 10th Workshop on Planning, Perception and Navigation for Intelligent Vehicles, Madrid, Spain, october 2018. https://hal.archives-ouvertes.fr/hal-01903327

Different tests are computed to obtain a more robust relative pose, they are described in:

Héry E, Xu Ph, Bonnifait Ph. Consistent Decentralized Cooperative Localization for Autonomous Vehicles using LiDAR, GNSS and HD maps. J Field Robotics. 2020;0-0. https://doi.org/10.1002/rob.22004

This post process data are only available in the CSV datasets.

5 Datasets

5.1 Rosbag dataset

The following ROS messages were recorded from each vehicle:

From the RTK GNSS/INS receiver Novatel SPAN CPT:

novatel_msgs/INSPVAX (50 Hz)[1]span/enuposeCovs

novatel_msgs/CORRIMUDATA (50 Hz)[2]

From the mono-frequency GNSS receiver Ublox M8T EVK:

ublox_msgs/NavPVT (2 Hz)[3]

From the LiDAR SICK LDMRS:

sensor_msgs/PointCloud2 (12.5 Hz)[4]

From the LiDAR SICK LMS 151:

sensor_msgs/LaserScan (50 Hz) [5]

From the novatel_msgs/INSPVAX and the ublox_msgs/NavPVT, geometry_msgs/PoseWithCovarianceStamped were computed to give the poses of the bases for these GNSS receivers.[6]

From the CAN bus messages a geometry_msgs/TwistWithCovarianceStamped were computed to give the speed and yaw rate of the vehicles.[7]

Finally, the HD map composed of the center of the lanes and the lane borders as well as the trajectory (the center of the lanes followed by the vehicles) were recorded with visualization_msgs/MarkerArray [8] and geometry_msgs/PolygonStamped[9]

Topic	Type
/map_markers/map_markers	visualization_msgs/MarkerArray
/map_markers/trajectory	geometry_msgs/PolygonStamped
/tf	tf2_msgs/TFMessage
/zoe_{g/w}/ldmrs/cloud	sensor_msgs/PointCloud2
/zoe_w/lms/scan	sensor_msgs/LaserScan
/zoe_{g/w}/span/enuposeCovs	geometry_msgs/PoseWithCovarianceStamped
/zoe_{g/w}/span/novatel_data/corrimudata	novatel_msgs/CORRIMUDATA
/zoe_{g/w}/span/novatel_data/inspvax	novatel_msgs/INSPVAX
/zoe_{g/w}/ublox/enuposeCovs	geometry_msgs/PoseWithCovarianceStamped
/zoe_{g/w}/ublox/navpvt	ublox_msgs/NavPVT
/zoe_{g/w}/vehicle/twist	geometry_msgs/TwistWithCovarianceStamped

Figure 8: Summary of the different ROS topics with correspondent types recorded in the ROS bag dataset.

/zoe_{g/w} corresponds to /zoe_g, the gray Renault Zoé (the follower) or /zoe_w, the white Renault Zoé (the leader).

[1] https://github.com/ros-drivers/novatel_span_driver/blob/master/novatel_msgs/msg/INSPVAX.msg
https://docs.novatel.com/oem7/Content/SPAN_Logs/INSPVAX.htm

[2] https://github.com/ros-drivers/novatel_span_driver/blob/master/novatel_msgs/msg/CORRIMUDATA.msg
https://docs.novatel.com/OEM7/Content/SPAN_Logs/CORRIMUDATA.htm

[3] https://docs.ros.org/noetic/api/ublox_msgs/html/msg/NavPVT.html
https://www.u-blox.com/en/docs/UBX-14041540

[4] https://docs.ros.org/noetic/api/sensor_msgs/html/msg/PointCloud2.html

[5] https://docs.ros.org/noetic/api/sensor_msgs/html/msg/LaserScan.html

[6] https://docs.ros.org/noetic/api/geometry_msgs/html/msg/PoseWithCovarianceStamped.html

[7] https://docs.ros.org/noetic/api/geometry_msgs/html/msg/TwistWithCovarianceStamped.html

[8] https://docs.ros.org/noetic/api/visualization_msgs/html/msg/MarkerArray.html

[9] https://docs.ros.org/noetic/api/geometry_msgs/html/msg/PolygonStamped.html

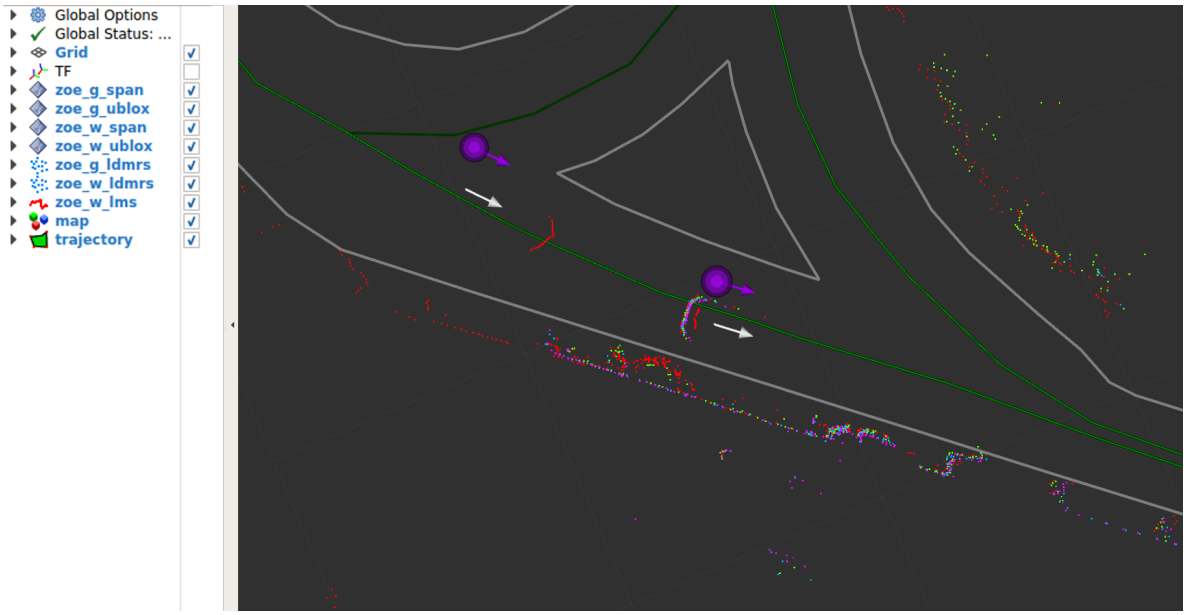


Figure 9: rviz display of the ROS bag dataset.

To visualize this dataset on rviz (see figure 9) the following command can be executed:

```
roslaunch static_tf_coop.launch
rosrun rviz rviz -d coop.rviz
rosbag play coop.bag
```

5.2 CSV dataset

The following .csv files were generated from the ROS messages:

Kinetics vehicle data from ESP:

Python example:

```
import pandas as pd
df = pd.read_csv('follower_kinetics.csv')
#df = pd.read_csv('leader_kinetics.csv')
df['time'] # time (s)
df['lon_vel'] # longitudinal speed (m/s)
df['yaw_rate'] # yaw rate (rad/s)
df['lon_acc'] # longitudinal acceleration (m/s^2)
df['lat_acc'] # lateral acceleration (m/s^2)
```

RTK GNSS/INS receiver Novatel SPAN CPT:

novatel_msgs/INSPVAX (50 Hz)

Python example:

```
import pandas as pd
df = pd.read_csv('follower_gnss_ref.csv')
#df = pd.read_csv('leader_gnss_ref.csv')
df['time'] # time (s)
df['x'] # east (m)
df['y'] # north (m)
df['z'] # up (m)
df['x_std'] # east (m)
df['y_std'] # north (m)
df['z_std'] # up (m)
df['roll'] # roll (rad)
df['pitch'] # pitch (rad)
df['yaw'] # yaw (rad)
df['roll_std'] # roll standard deviation (rad)
df['pitch_std'] # pitch standard deviation (rad)
df['yaw_std'] # yaw standard deviation (rad)
df['lat'] # latitude (deg)
df['lon'] # longitude (deg)
df['height'] # height above ellipsoid (m)
df['alt'] # height above mean sea level (m)
df['vel_x'] # east velocity (m/s)
df['vel_y'] # north velocity (m/s)
df['vel_z'] # up velocity (m/s)
df['vel_x_std'] # east velocity standard deviation (m/s
df['vel_y_std'] # north velocity standard deviation (m/s)
df['vel_z_std'] # up velocity standard deviation (m/s)
```

novatel_msgs/CORRIMUDATA (50 Hz)
Python example:

```
import pandas as pd
df = pd.read_csv('follower_imu_ref.csv')
#df = pd.read_csv('leader_imu_ref.csv')
df['time'] # time (s)
df['roll_rate'] # roll rate (rad/s)
df['pitch_rate'] # pitch rate (rad/s)
df['yaw_rate'] # yaw rate (rad/s)
df['lon_acc'] # Longitudinal acceleration (m/s^2)
df['lat_acc'] # Lateral acceleration (m/s^2)
df['vert_acc'] # Vertical acceleration (m/s^2)
```

Mono frequency GNSS receiver Ublox M8T EVK:
ublox_msgs/NavPVT (2 Hz)
Python example:

```
import pandas as pd
df = pd.read_csv('follower_gnss.csv')
#df = pd.read_csv('leader_gnss.csv')
df['time'] # time (s)
df['time_acc'] # time accuracy (s)
df['x'] # east (m)
df['y'] # north (m)
df['bias_x'] # east bias (m)
df['bias_y'] # north bias (m)
df['z'] # up (m)
df['lat'] # latitude (deg)
df['lon'] # longitude (deg)
df['height'] # height above ellipsoid (m)
df['alt'] # height above mean sea level (m)
df['h_acc'] # horizontal accuracy (m)
df['v_acc'] # vertical accuracy (m)
df['dop'] # position Dilution Of Precision (1)
df['yaw'] # yaw (heading in the ENU frame) (rad)
df['yaw_acc'] # yaw accuracy (rad)
df['vel_x'] # east velocity (m/s)
df['vel_y'] # north velocity (m/s)
df['vel_z'] # up velocity (m/s)
df['vel_h'] # horizontal/ground velocity (m/s)
df['vel_acc'] # velocity accuracy (m/s)
```

SICK LMS 151 LiDAR:
Python example:

```
import pandas as pd
df = pd.read_csv('leader_back_lidar.csv')
df['time'] # time (ns)
df['angle_min'] # min angle (rad)
df['angle_max'] # max angle (rad)
df['angle_increment'] # angular step between two angles (rad)
df['range_min'] # min range (m)
df['range_max'] # max range (m)
df['range0'] # range 0 (m)
...
df['range540'] # range 540 (m)
df['intensity0'] # light intensity 0
...
df['intensity540'] # light intensity 540
```

Point to line iterative closest point (PLICP):
Python example:

```
import pandas as pd
df = pd.read_csv('follower_plicp.csv')
Point to line iterative closest point (PLICP)
df['time'] # time (s)
df['x'] # relative longitudinal coordinate (m)
df['y'] # relative lateral coordinate (m)
df['yaw'] # relative yaw (rad)
df['cov_x'] # covariance matrix of the relative pose (m^2)
df['cov_y'] # covariance matrix of the relative pose (m^2)
df['cov_yaw'] # covariance matrix of the relative pose (rad^2)
df['cov_xy'] # covariance matrix of the relative pose (m^2)
df['cov_xyaw'] # covariance matrix of the relative pose (m*rad)
df['cov_yyaw'] # covariance matrix of the relative pose (m*rad)
df['usable'] # usable flag to filter bad estimation
df['mean_point2model_dist'] # Mean distance between the points and the model (m)
df['plicp_cond'] # condition number of the PLICP
df['plicp_det'] # determinant of the PLICP
df['khi2_pos'] # pose mahalanobis distance with the initial pose
df['khi2_yaw'] # yaw mahalanobis distance with the initial yaw
df['cluster_nb_points'] # number of points in the cluster
df['box_x_dpos'] # correction computed from the bounding box (m)
df['box_y_dpos'] # correction computed from the bounding box (m)
df['box_x_center'] # center of the bounding box (m)
df['box_y_center'] # center of the bounding box (m)
df['box_with'] # with of the bounding box (m)
df['box_lenght'] # lenght of the bounding box (m)
df['x_init'] # initial relative longitudinal coordinate (m)
df['y_init'] # initial relative lateral coordinate (m)
df['yaw_init'] # initial relative yaw (rad)
```

5.3 PCD LiDAR dataset

PCD (12.5 Hz)[10] files from the PCL library are used for the SICK LDMRS LiDAR datasets:

[10] https://pcl.readthedocs.io/projects/tutorials/en/latest/pcd_file_format.html

Python example:

```
import pcl
cloud = pcl.load('file.pcd') # LiDAR points scan (m)
```

5.4 Synchronized CSV dataset

Datasets were generated from the CSV datasets synchronized at the times of the PLICP dataset at 12.5 Hz. A Matlab/Octave reader is present to tests this data.