UNIVERSITY OF TORONTO
INSTITUTE FOR AEROSPACE STUDIES

December 13, 2020

A Guide to the Boreas Dataset

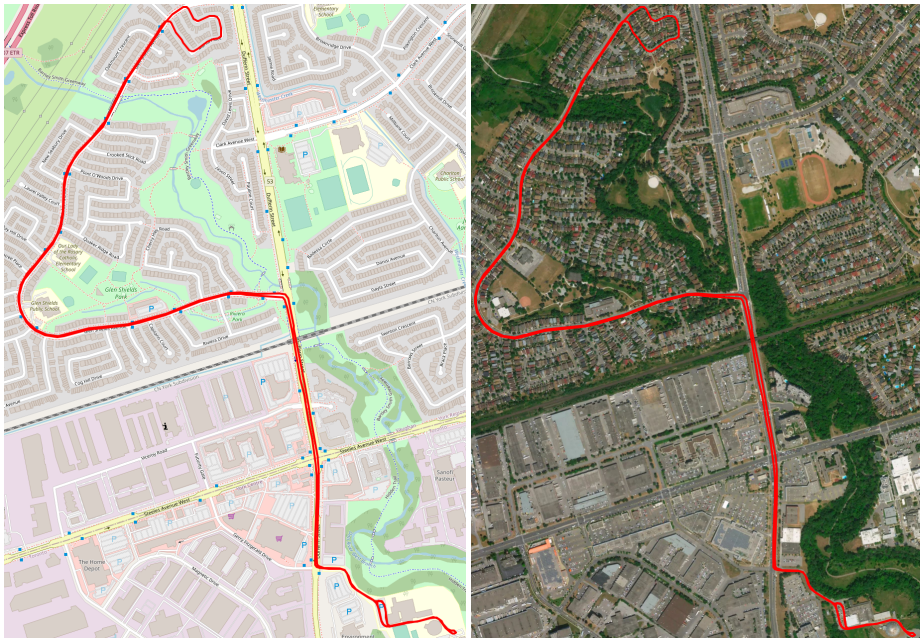
<keenan.burnett@utoronto.ca>

1 Introduction

Welcome, and thank you for expressing interest in using our dataset.

What is contained in this dataset? This dataset contains lidar data from a 128-beam Velodyne Alpha-Prime, radar data from a Navtech CIR204-H, camera data from a single FLIR Blackfly S (5 MP) monocular camera, and post-processed GNSS positioning data obtained from our Applanix POS LV system. All data was collected while driving a repeated route in Toronto, across several seasons and weather. The route is shown in Figure 1.

What is this dataset for? The intended purpose of this dataset is to explore radar-based and lidar-based localization techniques which including frame-to-frame odometry and metric mapping and localization. Eventually, we intend to add sequences which cover several weather and seasonal variations. Due to its longer wavelength, we expect radar-based maps to persist across seasons better than lidar-based maps. Further, we expect radar-based localization to perform better under rain and snow. This dataset will allow both of these hypotheses to be tested.



(a) OpenStreetMap Data (b) Mapbox Satellite Data

Figure 1: The Glen Shields route in Toronto, Ontario, Canada.

2 Data Organization

Each sequence is stored as a separate S3 bucket and follows the same naming convention: `boreas-YYYY-MM-DD-HH-MM` denoting the time (EDT) at which the data started to be collected. Figure 2 provides an overview of the folder structure of each sequence and S3 bucket. We extract camera images at 10 Hz, and extract each lidar pointcloud and radar scan. Note that we provide the original rosbags for those interested in working with them directly. Although, it is important to note that the positioning data is much less than that the post-processed data. Please note that you only need to download the data you want. Downloading the rosbags in addition to the extracted data would be redundant.

```
boreas-YYYY-MM-DD-HH-MM
├── applanix
│   ├── camera_event_time.csv
│   ├── camera_poses.csv
│   ├── gps_post_process.csv
│   ├── imu.csv
│   ├── lidar_poses.csv
│   ├── radar_poses.csv
│   └── ros_and_gps_time.csv
├── calib
│   ├── T_sens1_sens2.txt
│   └── ...
├── camera
│   └── <timestamp>.png
├── lidar
│   └── <timestamp>.bin
├── radar
│   └── <timestamp>.png
└── rosbags
    └── <timestamp>.bag
```

Figure 2: Folder of each sequence and S3 bucket.

3 Timestamps

The timestamp indicated in the name of each sensor file is a ROS-based time stamp. We have several betters methods for extracting an accurate timestamp for each sensor that is synchronized with GPS time. This is important for obtaining accurate positioning information for each sensor. For cameras, we record the GPS time at which exposure started in `camera_event_time.csv`. For lidar pointclouds, each point cloud is saved to a binary file of floats where each point is represented by $[x\ y\ z\ i\ r\ t]$ where (x, y, z) is the position of the point with respect to the lidar, i represents the infrared reflectivity, r represents the laser number of the velodyne, and t is a GPS time stamp. In order to time stamp radar measurements, we run an NTP server on Boreas which is synchronized with GPS time. A timestamp for each azimuth of radar scans is recorded.

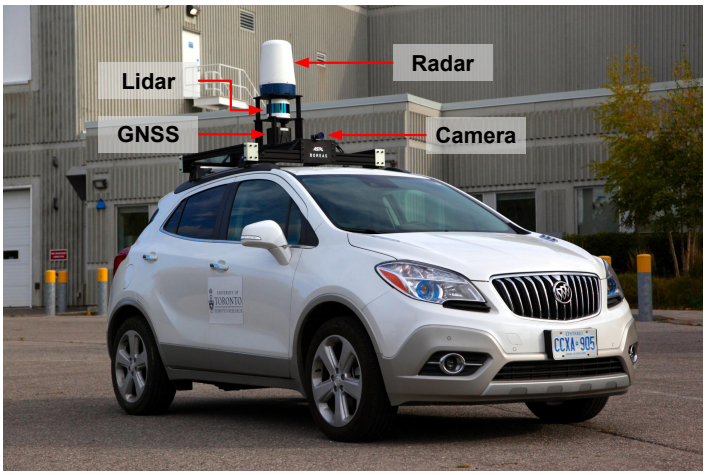
Note that we follow Oxford's convention and embed timestamp and encoder information into the first 11 columns of each polar radar scan. Raw radar scans are 2D polar images: N azimuths x R range bins. The first 8 columns correspond to an EPOCH time stamp associated with the NTP server. The next 2 columns represent the encoder value and can be converted into an azimuth angle for the scanner. The last column is a *valid* bit which we do not use but preserve for compatibility with the Oxford format.

4 Calibration

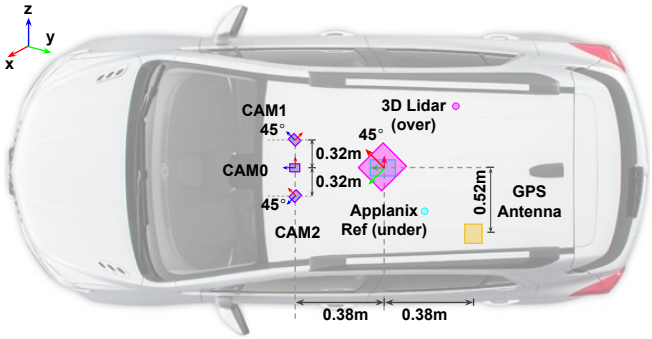
The camera intrinsics are calibrated using MATLAB's camera calibrator [1]. The extrinsic calibration between the camera and lidar is obtained using MATLAB's camera to LIDAR calibrator [2]. To calibrate the extrinsics between the lidar and radar, we use correlative scan matching via the Fourier Mellin transform [3]. Extrinsic calibrations are provided as 4x4 homogeneous transformation matrices: $\mathbf{T}_{ba} = \begin{bmatrix} \mathbf{C}_{ba} & \mathbf{r}_b^{ab} \\ \mathbf{0}^T & 1 \end{bmatrix} \in SE(3)$

TODO: calibrate lidar to applanix reference frame.

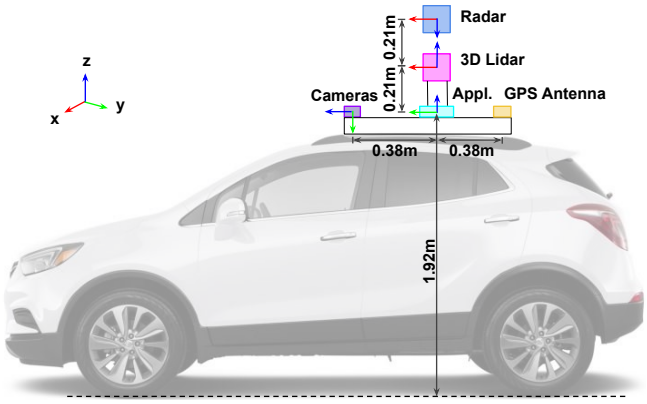
5 Boreas Sensor Frames



(a) Boreas



(b) Top View



(c) Side View

Figure 3: These diagrams of sensor locations and orientations are provided as a quick reference only. We have provided hand-measured and calibrated transformation matrices for several pairs of sensor frames.

In this document, we follow the convention used at UTIAS for describing rotations, translations, and transformation ma-

trices. $\mathbf{r}_a^{ba}$ refers to the translation of frame b with respect to frame a as measured in frame a. $\mathbf{C}_{ba} \in SO(3)$ is a 3x3 rotation matrix such that $\mathbf{r}_b = \mathbf{C}_{ba}\mathbf{r}_a$, $\mathbf{C}^T\mathbf{C} = \mathbf{I}$ and det$\mathbf{C} = 1$. Transformations are then defined as 4x4 homogeneous transformation matrices: $\mathbf{T}_{ba} = \begin{bmatrix} \mathbf{C}_{ba} & \mathbf{r}_b^{ab} \\ \mathbf{0}^T & 1 \end{bmatrix} \in SE(3)$.

Note 1: $\mathbf{T}_{ba}$ transforms a vector from frame a to point c as defined in a $\mathbf{r}_a^{ca}$ into $\mathbf{r}_b^{cb}$ ($\mathbf{r}_b^{cb} = \mathbf{T}_{ba}\mathbf{r}_a^{ca}$). Transformations like this can be used to transform lidar points into the camera's frame and so on. See (Barfoot, 2017) [4] for more details.

Note 2: For the sake of extreme clarity, $\mathbf{T}_{ba}$ is the transformation from frame a to frame b. Note the order of the indices here as well as for translations and rotations.

6 Applanix Measurement Frames

Raw GPS position measurements are provided as Latitude, Longitude, and Altitude (**LLA**). These measurements are provided in the **WGS84** (World Geodetic System 1984 Revision) standard. "It comprises a standard coordinate frame for the Earth, a standard spheroidal reference surface (the *datum* or *reference ellipsoid*) for raw altitude data, and a gravitational equipotential surface (the geoid) that defines the nominal sea level." [5]

The Applanix has two virtual measurement frames: the **Applanix reference frame** and the **vehicle frame**, and one physical sensor frame: the **IMU frame**. The orientation of the vehicle frame is defined as x-forwards, y-right, and z-down. The Applanix reference frame will be used to provide position and orientation data with respect to the origin. Nominally, the Applanix reference frame and vehicle frame have the same position and orientation (x-forwards, z-down). However, if desired, the Applanix reference frame can be assigned a **mounting angle** which differs from the vehicle frame orientation. We do this for Boreas in order to obtain an Applanix reference frame with x-right, y-forwards, and z-up as shown below:

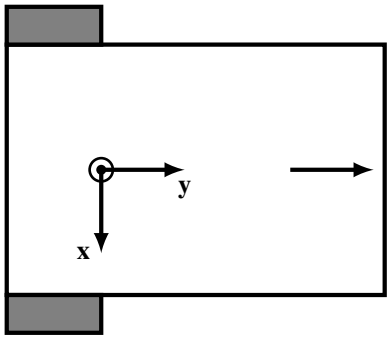


Figure 4: Applanix Reference Frame.

Note 1: The position of the Applanix reference frame and vehicle frame are defined with respect to the IMU frame. The IMU frame is the physical mounting position and orientation of the IMU sensor on the robot. For simplicity, we choose to have our virtual Applanix reference frame in the same position as the IMU frame.

Note 2: Position, orientation, and velocity information are provided about the Applanix reference frame, but the body-centered linear acceleration and angular velocity are provided in the IMU frame which has x-forwards, y-right, and z-down.

7 Applanix Position Data

7.1 GPS Raw

"Raw" GPS position data is referred to as such because it is the output of the GPS/IMU system as it was received in real-time. No post-processing is applied. As such, the accuracy can vary from 5-20 cm of error with real-time corrections or up to 3.0 m without real-time corrections. Locally, the positioning accuracy may be good enough for testing perception algorithms like object tracking, but this accuracy is insufficient for benchmarking odometry algorithms or constructing a map. If you play back a rosbag that was collected while driving, the `\navsat\odom` topic contains the raw position information.

Usually, we need to convert from latitude-longitude-altitude into a metric coordinate frame. **UTM** (Universal Transverse Mercator) coordinates are used for this purpose. UTM divides the earth into 60 zones and projects each to the plane as a basis for its coordinates [6]. Converting from latitude-longitude to UTM outputs a metric value for Easting and Northing.

The next question is how to orient the origin frame with respect to the Easting-Northing cardinal directions. Applanix prefers to use x-north, y-east, z-down which is abbreviated to **NED**. However, we prefer to use x-east, y-north, z-up which is abbreviated to **ENU**. Our GPS Raw position, orientation, and velocity information is reported for the Applanix reference frame with respect to ENU as show below:

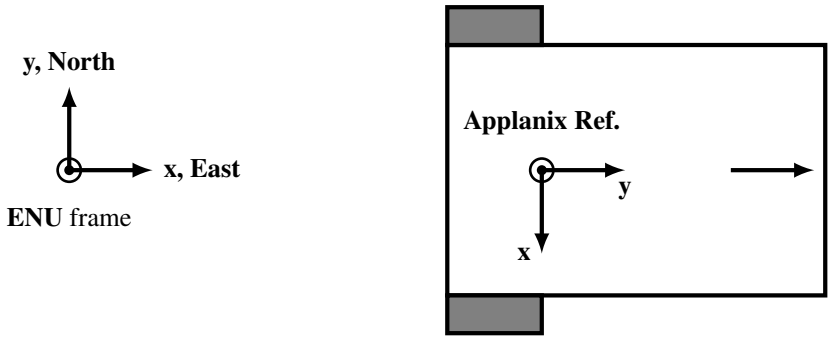


Figure 5: Applanix Reference Frame and ENU (East-North-Up).

Note 1: Not all GPS data sources use the same datum for reporting altitude. If you stick to producing and consuming data from the exact same sensor, then you shouldn't have any issues. However, if you want to localize within a map created by a different company's sensor, you may run into problems. You may end up with a static z offset between the reference frames. The easiest way to check for is to move your sensor to a known location in the map and then check the difference in z values.

Note 2: When using ENU, the heading is zero when the vehicle is pointing North, and a positive heading is defined when the vehicle rotates counter-clockwise.

7.2 GPS Post-Processed

Post-processed GPS results incorporate all available data (GPS, IMU, differential corrections) and perform a batch optimization (using an RTS smoother) to obtain the most accurate positioning information at each time step. Without differential corrections, the accuracy can be 5-15 cm. With differential corrections, the accuracy can be 1-5 cm or less. By default, post-processed position, orientation, and velocity information is reported as Applanix reference frame with respect to NED.

Note that the heading, and velocity reported is with respect to a local frame that contains a "wander angle". To fix this, we first need to extract the true heading:

$$\Psi_{true} := \Psi_{wander} - \alpha \tag{1}$$

Ψ_{wander} is the heading that's reported in the post-processed output. α is also reported as the "wander angle". Note that we also need to transform the velocities from this wander frame into what we actually want:

$$V_N = V_x \cos\alpha - V_y \sin\alpha \tag{2}$$
$$V_E = -V_x \sin\alpha - V_y \cos\alpha \tag{3}$$

Note that our desired linear velocities in ENU are then $V_x := V_E$, $V_y := V_N$, and V_z remains unchanged. In order to convert the orientation which is reported with respect to NED into an orientation reported with respect to ENU, we use the following conversions which are explained more thoroughly in sections 8.3, 8.4.

$$\mathbf{C}_{ned,applanix} = \mathbf{C}_1(\text{roll}_{ned})\mathbf{C}_2(\text{pitch}_{ned})\mathbf{C}_3(\text{heading}_{ned}) \tag{4}$$
$$\mathbf{C}_{enu,ned} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{5}$$
$$\mathbf{C}_{enu,applanix} = \mathbf{C}_{enu,ned}\mathbf{C}_{ned,applanix} \tag{6}$$

We can then use our conversion in section 8.4 to convert $\mathbf{C}_{enu,applanix}$ into heading_{enu}, pitch_{enu}, roll_{enu}. Finally, to convert the UTM Northing and Easting into a position in ENU, we use the following assignments:

$$x_{enu} := \text{Easting} \tag{7}$$
$$y_{enu} := \text{Northing} \tag{8}$$
$$z_{enu} := \text{Altitude} \tag{9}$$

8 Useful Conversions

8.1 Quaternion to 3x3 Rotation

GPS Raw poses are reported as a ROS quaternion $\mathbf{q} = [q_x\ q_y\ q_z\ q_w]^T$ and a position $\mathbf{r}_e^{re} = [x\ y\ z]^T$. $\mathbf{r}_e^{re}$ describes the position of the Applanix reference frame with respect to the ENU frame, as reported in the ENU frame. It is useful to convert this into a 4x4 homogeneous transformation matrix. We will denote this 4x4 transformation matrix from the Applanix reference frame r to the ENU frame e as [4]:

$$\mathbf{T}_{er} = \begin{bmatrix} \mathbf{C}_{er} & \mathbf{r}_e^{re} \\ \mathbf{0}^T & 1 \end{bmatrix}. \tag{10}$$

In order to obtain the 3x3 rotation matrix $\mathbf{C}_{er}$, we use the following formula for working with the ROS quaternion [4]:

$$\mathbf{C}_{er} = (\eta^2 - \boldsymbol{\xi}^T\boldsymbol{\xi})\mathbf{1} + 2\boldsymbol{\xi}\boldsymbol{\xi}^T - 2\eta\boldsymbol{\xi}^\times, \tag{11}$$

where $\boldsymbol{\xi} = [q_x\ q_y\ q_z]^T$, $\eta = q_w$, and $(\cdot)^\times$ is the skew-symmetric operator [4]:

$$\mathbf{u}^\times = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}^\times := \begin{bmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{bmatrix}. \tag{12}$$

8.2 3x3 Rotation to Quaternion

Using the axis-angle representation for a rotation, we can represent a rotation by a unit-length axis **a** and a single rotation angle ϕ, 4 parameters and 1 constraint ($\mathbf{a}^T\mathbf{a} = 1$). Note that $\mathbf{Ca} = \mathbf{a}$ which implies that **a** is a (unit-length) eigenvector of **C**. The angle can be found by exploiting the trace of a rotation matrix [4]:

$$\phi = \cos^{-1}\Big(\frac{tr(\mathbf{C}) - 1}{2}\Big) + 2\pi m \tag{13}$$

A unit-length quaternion can then be generated using the following formulas:

$$\boldsymbol{\xi} = \mathbf{a}\sin\frac{\phi}{2} \tag{14}$$

$$\eta = \cos\frac{\phi}{2} \tag{15}$$

$$\mathbf{q} = \begin{bmatrix}\boldsymbol{\xi}\\ \eta\end{bmatrix} \tag{16}$$

Note that the space of quaternions is a double cover of SO(3), so each 3x3 rotation matrix maps to two quaternions: $\pm\mathbf{q}$.

8.3 Yaw, Pitch, Roll to 3x3 Rotation

The post-processed GPS provides the heading (yaw), pitch, and roll of the Applanix reference frame with respect to NED which we convert into yaw, pitch, roll values with respect to ENU. To convert this into a 3x3 rotation matrix, we use the following formula based on the Tait-Bryan sequence of rotations [7]:

$$\mathbf{C}_{er} = \mathbf{C}_1(roll)\mathbf{C}_2(pitch)\mathbf{C}_3(yaw), \tag{17}$$

where C_1, C_2, C_3 are the principal rotation matrices about x, y, and z respectively:

$$\mathbf{C}_1(\theta_1) = \begin{bmatrix}1 & 0 & 0\\ 0 & \cos\theta_1 & \sin\theta_1\\ 0 & -\sin\theta_1 & \cos\theta_1\end{bmatrix} \tag{18}$$

$$\mathbf{C}_2(\theta_2) = \begin{bmatrix}\cos\theta_2 & 0 & -\sin\theta_2\\ 0 & 1 & 0\\ \sin\theta_2 & 0 & \cos\theta_2\end{bmatrix} \tag{19}$$

$$\mathbf{C}_3(\theta_3) = \begin{bmatrix}\cos\theta_3 & \sin\theta_3 & 0\\ -\sin\theta_3 & \cos\theta_3 & 0\\ 0 & 0 & 1\end{bmatrix} \tag{20}$$

$$\tag{21}$$

8.4 3x3 Rotation to Yaw, Pitch, Roll

$$c_y = \sqrt{\mathbf{C}(2,2)^2 + \mathbf{C}(1,2)^2} \tag{22}$$

if $c_y > \epsilon$ (where ϵ is chosen to be small such as 1×10^{-15}) then

$$\text{yaw} = \text{atan2}(\mathbf{C}(0,1), \mathbf{C}(0,0)) \tag{23}$$
$$\text{pitch} = \text{atan2}(-\mathbf{C}(0,2), c_y) \tag{24}$$
$$\text{roll} = \text{atan2}(\mathbf{C}(1,2), \mathbf{C}(2,2)) \tag{25}$$

else if $c_y \leq \epsilon$,

$$\text{yaw} = \text{atan2}(-\mathbf{C}(1,0), \mathbf{C}(1,1)) \tag{26}$$
$$\text{pitch} = \text{atan2}(-\mathbf{C}(0,2), c_y) \tag{27}$$
$$\text{roll} = 0 \tag{28}$$

References

[1] "Single camera calibrator app," https://www.mathworks.com/help/vision/ug/single-camera-calibrator-app.html, accessed: 2020-12-13.

[2] "Lidar and camera calibration," https://www.mathworks.com/help/lidar/ug/lidarcameracalibrationexample.html, accessed: 2020-12-13.

[3] "radar to lidar calib," https://github.com/keenan-burnett/radar_to_lidar_calib, accessed: 2020-12-13.

[4] T. D. Barfoot, *State estimation for robotics*. Cambridge University Press, 2017.

[5] "WGS84," http://wiki.gis.com/wiki/index.php/WGS84, accessed: 2020-11-23.

[6] "Universal transverse mercator coordinate system," https://en.wikipedia.org/wiki/Universal_Transverse_Mercator_coordinate_system, accessed: 2020-11-24.

[7] "Euler angles: Tait-Bryan angles," https://en.wikipedia.org/wiki/Euler_angles#Tait%E2%80%93Bryan_angles, accessed: 2020-11-24.