i.c.sens Visual-Inertial-LiDAR Dataset

Raphael Voges

August 18, 2020

Abstract

This documentation describes the i.c.sens Visual-Inertial-LiDAR Dataset, a data set for the evaluation of dead reckoning or SLAM approaches in the context of mobile robotics. It consists of street-level monocular RGB camera images, a front-facing 180° point cloud, angular velocities, accelerations and an accurate ground truth trajectory. In total, we provide around 77 GB of data resulting from a 15 minutes drive, which is split into 8 rosbags of 2 minutes (10 GB) each. Besides, the intrinsic camera parameters and the extrinsic transformations between all sensor coordinate systems are given. In the following, we give a detailed overview and explanation of the data acquisition, the post-processing steps and the data format in general. The data is related to several publications [1–8].

1 Data Acquisition

The data set was acquired in the context of the measurement campaign described in [9]. Here, a vehicle, which can be seen in Figure 1, was equipped with a self-developed sensor platform and a commercially available Riegl VMX-250 Mobile Mapping System. This Mobile Mapping System consists of two laser scanners, a camera system and a localization unit containing a highly accurate GNSS/IMU system.

The data acquisition took place in May 2019 during a sunny day in the Nordstadt of Hannover (coordinates 52.388598, 9.716389). The route we took can be seen in Figure 2. This route was completed three times in total, which amounts to a total driving time of 15 minutes.

Figure 1: The measurement vehicle with the self-developed sensor platform in the front and the commercially available Mobile Mapping System in the back. Image credit: Sören Vogel.
Figure 2: The route we took for our measurement campaign.

Table 1: Properties of the RGB camera and the corresponding images.

<table>
<thead>
<tr>
<th>Camera</th>
<th>Pointgrey (FLIR) GS3-U3-23S6C-C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lense</td>
<td>Tamron M111FM08</td>
</tr>
<tr>
<td>Focal length</td>
<td>8.0 mm</td>
</tr>
<tr>
<td>Image size</td>
<td>1920 px × 1200 px</td>
</tr>
<tr>
<td>Pixel size</td>
<td>5.86 µm</td>
</tr>
<tr>
<td>Frame rate</td>
<td>10 FPS</td>
</tr>
<tr>
<td>Exposure time</td>
<td>1 ms</td>
</tr>
</tbody>
</table>

1.1 Sensor platform

The self-developed sensor platform consists of several sensors. This documentation focuses on the following sensors for which data is provided in the published data set:

- Velodyne HDL-64 LiDAR
- LORD MicroStrain 3DM-GQ4-45 GNSS aided IMU
- Pointgrey GS3-U3-23S6C-C RGB camera

LiDAR  The laser scanner was set to rotate with a frequency of 10 Hz resulting in a horizontal angular resolution of 0.1728°. It consists of 64 laser beams firing simultaneously that are arranged on top of each other resulting in a vertical angular resolution of 0.5°. Thus, the Velodyne HDL-64E laser scanner delivers a vertical field of view from 2° to −29.5°. The right hand coordinate system of the LiDAR is defined as follows. The x-axis is facing forward in driving direction, the y-axis is pointing to the left and the z-axis is pointing upwards.

Camera  The properties of the camera are detailed in Table 1. It is set to capture images whenever the laser scanner faces forward, thus resulting in a frame rate of 10 frames per second. More details on this synchronization can be found in Section 1.3. The right hand coordinate system of the camera is defined as follows. The x-axis is pointing to the right, the y-axis is pointing downwards and the z-axis is pointing forward in driving direction.
Inertial Measurement Unit (IMU)  The IMU was operated at a measurement frequency of 500 Hz. The right hand coordinate system of the IMU is defined as follows. The x-axis is pointing forward in driving direction, the y-axis is pointing to the right and the z-axis is pointing downwards. Table 2 shows the most important specifications of the IMU.

<table>
<thead>
<tr>
<th></th>
<th>Accelerometer</th>
<th>Gyroscope</th>
</tr>
</thead>
<tbody>
<tr>
<td>Measurement range</td>
<td>±5 g</td>
<td>300°/s</td>
</tr>
<tr>
<td>Non-linearity</td>
<td>±0.03 % fs</td>
<td>±0.03 % fs</td>
</tr>
<tr>
<td>Resolution</td>
<td>&lt;0.04 mg</td>
<td>&lt;0.0025°/s</td>
</tr>
<tr>
<td>Bias instability</td>
<td>±0.02 mg</td>
<td>5°/h</td>
</tr>
<tr>
<td>Initial bias error</td>
<td>±0.001 g</td>
<td>±0.05°/s</td>
</tr>
<tr>
<td>Scale factor stability</td>
<td>±0.05 %</td>
<td>±0.05 %</td>
</tr>
<tr>
<td>Noise density</td>
<td>50 μg/√Hz</td>
<td>0.002°/s/√Hz</td>
</tr>
<tr>
<td>Alignment error</td>
<td>±0.05°</td>
<td>±0.05°</td>
</tr>
</tbody>
</table>

Extrinsic calibration

To determine the extrinsic transformation between all sensor coordinate systems, we conducted a precise indoor calibration. First, a platform coordinate system, which can be seen in Figure 5, was established for our self-developed sensor platform. Afterwards, we employed a laser tracker (Leica Absolute Tracker AT960) to measure reference points of the Microstrain IMU with a precision of less than 1 mm. Using the CAD model of the IMU, this allowed us to accurately determine the origin of the coordinate system of the IMU.

To determine the pose of the camera with respect to the platform coordinate system, we took an image of known 3D points in the calibration laboratory, which can be seen in Figure 5a. These 3D points were also measured using the laser tracker to determine their position with respect to the platform coordinate system. Then, the extrinsic transformation of the camera could be determined by computing its pose relative to these known 3D points, which requires to solve the Perspective-n-Point (PnP) problem.

Next, we computed the pose of the Velodyne LiDAR with respect to the camera, which then allowed us to determine the pose of the LiDAR with respect to the platform coordinate system. In order to do...
that, we employed a checkerboard, which can be seen in Figure 5b and solved the extrinsic calibration between camera and LiDAR using the approach presented in [7].

Finally, to determine the extrinsic transformation between the Mobile Mapping System on the back of the van and our self-developed sensor platform on the front of the van, we again employed the laser tracker. First, both sensor systems were rigidly mounted on the van, as can be seen in Figure 1. Afterwards, we measured reference points on both the Mobile Mapping System and our own platform to establish the extrinsic transformation between our self-defined sensor platform coordinate system and the Mobile Mapping System coordinate system defined by Riegl.

1.3 Time synchronization

For the time synchronization of camera and laser scanner, we employed the idea depicted in [10]. Here, the authors propose to use a reed contact attached to the rotating laser scanner to generate an electrical pulse that triggers the camera whenever the laser scanner is facing forward. Consequently, the camera and laser scanner are directly synchronized in time. In addition, the electrical pulse was also routed into a Raspberry Pi that is synchronized to GPS time using an EVK-M8T GNSS receiver and the GPS daemon (gpsd). Thus, we were able to assign an accurate GPS timestamp to every image and laser scan point. Since the IMU and the Riegl VMX-250 Mobile Mapping System encompass a GNSS receiver, the timestamps of their measured data are also given in the GPS time reference, and thus all our sensors are synchronized via GPS time.

Using this strategy, we are able to provide accurate timestamps for each and every sensor message with a synchronization accuracy of less than 1 ms.
Table 3: Content of the rosbags.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Total number of messages</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>/camera_right/image_raw</td>
<td>9379 msgs</td>
<td>sensor_msgs/Image</td>
</tr>
<tr>
<td>/camera_right/camera_info</td>
<td>9379 msgs</td>
<td>sensor_msgs/CameraInfo</td>
</tr>
<tr>
<td>/ground_truth</td>
<td>186818 msgs</td>
<td>geometry_msgs/PoseStamped</td>
</tr>
<tr>
<td>/imu/data</td>
<td>463333 msgs</td>
<td>sensor_msgs/Imu</td>
</tr>
<tr>
<td>/tf</td>
<td>186818 msgs</td>
<td>tf/tfMessage</td>
</tr>
<tr>
<td>/tf_static</td>
<td>934 msgs</td>
<td>tf/tfMessage</td>
</tr>
<tr>
<td>/velodyne_points</td>
<td>9378 msgs</td>
<td>sensor_msgs/PointCloud2</td>
</tr>
</tbody>
</table>

2 Description of the data and parameters

This section introduces all data and parameters that provided within this dataset. First, Section 2.1 describes the data gathered by our sensors. Second, Section 2.2 introduces the camera intrinsic parameters, which are required to use the camera images. Third, Section 2.3 explains the extrinsic calibration parameters determined during the calibration described in Section 1.2. Finally, Section 2.4 introduces a configuration file which can be used to display all relevant data in the 3D visualization tool for ROS: rviz.

2.1 Sensor data

The sensor data is provided exclusively in a rosbag, which is split into 8 consecutive rosbags each containing data from 2 minutes of driving. These rosbags contain the ROS topics detailed in Table 3. These rosbags can either be played back using

$ rosbag play icsens-visual-inertial-lidar-dataset-<number>.bag --clock$

or directly included into your program code using the rosbag API in python or C++.

In the following we explain the individual messages.

/camera_right/image_raw  The images have been anonymized using the Github project understand.ai Anonymizer. This means that faces and license plates have been blurred. Besides, the images are RGB encoded and can be processed using, for example, the ROS cv_bridge. The corresponding intrinsic camera parameters can be either found in the rosbag under the topic /camera_right/camera_info or in the supplemental camera_intrinsics.txt file. The images are given in the reference frame of the camera: "camera_right". The timestamps correspond to the start of the exposure, which is set to last 1 ms.

/camera_right/camera_info  The intrinsic parameters of the camera are provided simultaneously with each new image. More details on these can be found in Section 2.2.

/ground_truth  These messages contain the pose (3D position and orientation quaternion) of the vehicle relative to the start of the experiment. This data is gathered using the Riegl VMX-250 Mobile Mapping System and has a typical relative accuracy of a few centimeters [11]. The poses are given in the reference frame of the locally defined map, which corresponds to the first pose of the Mobile Mapping System: "map".

/imu/data  The IMU messages consist of the raw 3D acceleration and angular velocity measurements. In addition, an orientation (quaternion) is provided that is computed using an internal filtering algorithm of the Microstrain IMU. The corresponding covariance matrices are not provided but can be computed using Table 2 or additional information from the manufacturer. All measurements are given in the reference frame of the IMU: "imu".

5
These messages contain the same information as the /ground_truth topic and serve only to provide a transformation between the reference frame of the locally defined map "map" and the current frame of the Mobile Mapping System "mms". This can be employed to visualize the path of the vehicle with the corresponding images and point clouds in rviz.

These messages are published with a frequency of 1 Hz and contain the static transformations between all sensor coordinate systems. The transformation tree is set up as can be seen in Figure 6. More details on the extrinsic calibration process are provided in Section 1.2 and explanations on the calibration parameters can be found in Section 2.3.

These messages contain the forward-facing half (180°) of the point cloud of the Velodyne HDL-64E. The data was acquired using the ROS velodyne_driver that can be found on Github and in the ROS wiki. Thus, the PointCloud2 message can be converted into a point cloud containing X,Y,Z and intensity information for every point using the PCL library.

Each message corresponds to one full (360°) rotation of the LiDAR of which the back-facing half was omitted for this publication due to occlusions. In ROS, however, it is only possible to assign one common timestamp to the full point cloud. This timestamp corresponds to the moment in time when the laser scanner is facing to the front, i.e. the timestamp is only correct for points with a vertical opening angle of 0° (i.e. along the x-axis of the laser scanner). The accurate timestamp for all other points can be computed by considering the rotation speed of the LiDAR (600 rpm in clockwise direction), the laser firing sequence and the timing indicated in the sensor manual.

All measurements are given in the reference frame of the LiDAR: "velodyne".

The intrinsic parameters are provided in the rosbag under the topic /camera_right/camera_info and in the external calibration file camera_intrinsics.txt. Here, the $3 \times 3$ projection matrix, which is defined as follows, is stored in a row-wise order.

$$K = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$$

The corresponding transformation of a 3D point $(X \ Y \ Z)$ onto the image plane is thus defined as:

$$s \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = K (R \ | \ t) \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

where $s$ is the unknown scale factor, $(u, v)$ are the image coordinates in pixel and $(R \ | \ t)$ is the extrinsic transformation between the camera and the world.

Besides, the rosbag and the external calibration file contain information about the lens distortion. Here, we provide three radial distortion coefficients $k_1$, $k_2$, $k_3$ and two tangential distortion coefficients.
\(p_1, p_2\). These are stored as follows:

\[
D = \begin{pmatrix} k_1 & k_2 & p_1 & p_2 & k_3 \end{pmatrix}.
\]  

(3)

A description of each parameter and its usage can be found in the OpenCV documentation.

2.3 Extrinsic transformations

The extrinsic calibration parameters that have been found during the extrinsic calibration described in Section 1.2 are stored in the rosbag under the topic \texttt{/tf\_static} and in the external calibration file \texttt{extrinsic\_calibration\_parameters.txt}. They are defined as follows.

Each transformation between sensor coordinate systems is given as a 4 \(\times\) 4 transformation matrix \(T\). This matrix consists of the 3 \(\times\) 3 rotation matrix \(R\) and the 3 \(\times\) 1 translation vector \(t\):

\[
T = \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix}.
\]  

(4)

For example, the transformation matrix from the camera coordinate system "camera\_right" to the Mobile Mapping System coordinate system "mms" is denoted as \(T_1\). It allows to transform a 3D point \(X_C\) given in "camera\_right" into a 3D point \(X_M\) given in "mms":

\[
\begin{pmatrix} X_M \\ 1 \end{pmatrix} = T_1 \cdot \begin{pmatrix} X_C \\ 1 \end{pmatrix}
\]  

(5)

2.4 rviz configuration file

The external file \texttt{icsens\_data.rviz} provides a configuration file for the 3D visualization tool for ROS: rviz. rviz can be launched with this configuration file by executing:

\$ rosrun rviz rviz -d icsens\_data.rviz

By doing so, rviz directly displays the tf tree, the ground truth pose of the vehicle, the corresponding point cloud and the current camera image. Since the frame is fixed to the static frame "map" and all extrinsic transformations are provided, the movement of the vehicle with the corresponding point cloud can be observed.

3 Exemplary data

Finally, Figure 7, Figure 8 and Figure 9 provide some exemplary images and the corresponding point clouds from our dataset.

References


Figure 7: Exemplary images and corresponding point clouds from our dataset.
Figure 8: Exemplary images and corresponding point clouds from our dataset.
Figure 9: Exemplary images and corresponding point clouds from our dataset.