



**HAL**  
open science

## ConSLAM: Construction Data Set for SLAM

Maciej Trzeciak, Kacper Pluta, Yasmin Fathy, Lucio Alcalde, Stanley Chee,  
Antony Bromley, Ioannis Brilakis, Pierre Alliez

► **To cite this version:**

Maciej Trzeciak, Kacper Pluta, Yasmin Fathy, Lucio Alcalde, Stanley Chee, et al.. ConSLAM: Construction Data Set for SLAM. *Journal of Computing in Civil Engineering*, 2023, 37 (3), 10.1061/JCCEE5.CPENG-5212 . hal-04039392

**HAL Id: hal-04039392**

**<https://inria.hal.science/hal-04039392>**

Submitted on 21 Mar 2023

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Copyright

# ConSLAM: Construction Dataset for SLAM

Maciej Trzeciak<sup>1</sup>, Kacper Pluta<sup>2</sup>, Yasmin Fathy<sup>3</sup>, Lucio Alcalde<sup>4</sup>, Stanley Chee<sup>5</sup>, Antony Bromley<sup>6</sup>, Ioannis Brilakis<sup>7</sup>, and Pierre Alliez<sup>8</sup>

<sup>1</sup>Department of Engineering, University of Cambridge, CB2 1PZ, UK, mpt35@cam.ac.uk

<sup>2</sup>Inria Sophia Antipolis-Méditerranée, 06902 Valbonne, France, kacper.pluta@inria.fr

<sup>3</sup>Department of Engineering, University of Cambridge, CB2 1PZ, UK, yafa2@cam.ac.uk

<sup>4</sup>Laing O'Rourke, DA2 6SN, Dartford, UK, lalcalde@laingorourke.com

<sup>5</sup>Laing O'Rourke, DA2 6SN, Dartford, UK, schee@laingorourke.com

<sup>6</sup>Laing O'Rourke, DA2 6SN, Dartford, UK, abromley@laingorourke.com

<sup>7</sup>Department of Engineering, University of Cambridge, CB2 1PZ, UK, ib340@cam.ac.uk

<sup>8</sup>Inria Sophia Antipolis-Méditerranée, 06902 Valbonne, France, pierre.alliez@inria.fr

## ABSTRACT

This paper presents a dataset collected periodically on a construction site. The dataset aims to evaluate the performance of SLAM algorithms used by mobile scanners or autonomous robots. It includes ground-truth scans of a construction site collected using a terrestrial laser scanner along with five sequences of spatially registered and time-synchronized images, LiDAR scans and inertial data coming from our prototypical hand-held scanner. We also recover the ground-truth trajectory of the mobile scanner by registering the sequential LiDAR scans to the ground-truth scans and show how to use a popular software package to measure the accuracy of SLAM algorithms against our trajectory automatically. To the best of our knowledge, this is the first publicly accessible dataset consisting of periodically collected sequential data on a construction site.

## INTRODUCTION

Digitizing the geometry of an existing asset is a vital part of the *Digital Twin* (DT)

25 concept in the *Architecture, Engineering and Construction* (AEC) industry. A DT is a virtual  
26 representation of physical assets that mirrors their status and behaviour. The expanding use  
27 of mobile scanning technologies raises the prospect of more efficient data collection and,  
28 hence, improves generation of DTs. Mobile scanning systems create point clouds of scanned  
29 scenes faster than traditional workflows based on Terrestrial Laser Scanners (TLS). However,  
30 their underlying technology—called *SLAM* after *Simultaneous Localization And Mapping*—  
31 is inherently prone to accuracy-related problems, for example, due to drift when no Global  
32 Navigation Satellite System (GNSS) is integrated.

33 Several odometry and SLAM algorithms have been proposed in the last decade. However,  
34 they need to be even more accurate to meet the requirements of demanding use cases such  
35 as engineering surveying. There are already available datasets to evaluate the performance  
36 of such algorithms with the nuTonomy scenes (nuScenes) (Caesar et al. 2020) and Seman-  
37 ticPOSS (Pan et al. 2020), among others. Yet, only some are suitable for comparing SLAM  
38 methods on construction sites. To the best of our knowledge, only the Hilti SLAM challenge  
39 dataset (Helmberger et al. 2021) provides two sequences of data captured one after another  
40 on a construction site on the same day. However, they cannot show any progress done on  
41 site due to a short period of time between the recordings. Therefore, we can conclude that  
42 none of the publicly available datasets contains sequential data of the same construction  
43 site captured periodically. This would reflect real-world construction control use cases using  
44 hand-held scanners or autonomous robots.

45 To tackle this problem, we present a periodically collected real-world construction dataset,  
46 “ConSLAM”, accessible under the following link <https://github.com/mac137/ConSLAM>.  
47 Our dataset is designed to facilitate the performance measurement for odometry/SLAM  
48 algorithms on construction sites having in mind construction control tasks that are periodic.  
49 The SLAM community can now test how the performance of their algorithms changes along  
50 with the progress done on site. Also, our dataset can possibly open a new dimension to  
51 future SLAM algorithms which can leverage the fact that the scanned place is the same

52 but with certain changes resulting from the progress done over time. These are the main  
53 differences between our dataset and the mentioned earlier Hilti SLAM challenge.

54 This paper is an extension of our conference paper (Trzeciak et al. 2022), with the  
55 following new contributions: (1) we increase the coverage of the ground-truth trajectories  
56 from around 70% to around 98%; (2) we capture one more sequence of data in addition  
57 to the previously collected four of them; (3) we integrate a popular package for evaluating  
58 the trajectory of SLAM algorithms so that our dataset can be easily used among other  
59 known datasets, such as KITTI (Geiger et al. 2013) and TUM RGB-D (Sturm et al. 2012);  
60 (4) we show how to use our dataset on different SLAM and odometry algorithms and how  
61 to evaluate them against our ground-truth trajectory; (5) we show that the latest SLAM  
62 algorithms run on our dataset are still subject to drift which validates the need for this  
63 dataset and further research into SLAM algorithms with construction-related use cases in  
64 mind.

65 We used our prototypical hand-held scanner, *PointPix*, to create the ConSLAM, which  
66 includes five streams of data recorded every month on one of the floors of a construction  
67 site. Each sequence contains *Red-Green-Blue* (RGB) and *Near-InfraRed* (NIR) images of  
68 resolutions  $2064 \times 1544$  and  $2592 \times 1944$  pixels respectively, along with 16-beam Velodyne  
69 LiDAR scans and 9-axis *Inertial Measurement Unit* (IMU) data. The LiDAR, RGB and NIR  
70 sensors are synchronized in time and recorded at about 10 Hz while the IMU is recorded at  
71 about 400 Hz. The acquired sequences vary in their duration between 7-10 minutes. For  
72 every sequence, we also include a *Ground-Truth* (GT) point cloud for which a land surveying  
73 team used a Terrestrial Laser Scanner and accompanying software. We used these point  
74 clouds to recover the ground-truth trajectories of our scanner for sequences 2-5 with 98%  
75 coverage.

76 This paper is structured as follows. Section **EXISTING DATASETS** provides a discussion  
77 on the existing datasets. Section **METHODOLOGY** includes the description of our hand-  
78 held prototype and other devices and methods we used to produce the complete dataset.

79 In Section **DATASET: ConSLAM**, we briefly describe the construction site and present the  
80 structure as well as the availability of our dataset. We close the paper by discussing future  
81 steps in Section **CONCLUSION AND FUTURE DIRECTION**.

## 82 **EXISTING DATASETS**

83 Mobile scanners are portable devices that integrate multiple sensors whose  
84 streams/sequences of data can be recorded and used by odometry/SLAM algorithms. On  
85 one hand, commercially available hand-held scanners typically do not allow users to access  
86 these sequences. Instead, the user can access only the final scene-referenced/registered point  
87 cloud and, optionally, the trajectory of the scanner. Examples of such datasets can be seen  
88 in (Khoshelham et al. 2017). On the other hand, prototypical scanners built at research  
89 labs enable accessing streams of data whose publicly available sequential datasets can be  
90 classified into synthetic and real-world. A real-world dataset is collected from natural and  
91 real scenes, while a synthetic dataset is artificially generated in a virtual world by simulating  
92 a real-world data acquisition system. Although there are available sequential point cloud  
93 datasets, very few, such as the Hilti SLAM challenge dataset (Helmberger et al. 2021), are  
94 sequential and collected on construction sites. Yet, none of them are periodically collected,  
95 thus showing progress done on site. This would reflect real-world use cases in construction.  
96 Most studied sequential datasets are described in this section and summarized in Table 1.

97 **KITTI** is a benchmark primarily for 3D object detection scenarios (Geiger et al. 2012;  
98 Geiger et al. 2013). The dataset consists of six hours of traffic scenarios at 10–100 Hz with a  
99 system installed on a moving car travelling at a top speed of 90 km/h. The scanning system  
100 comprises high-resolution colour and grayscale stereo cameras, a Velodyne laser scanner, a  
101 GPS, and an IMU sensor (Geiger et al. 2013). The scanning system configuration enables  
102 data collection for a variety of applications, including stereo, optical flow, visual odometry  
103 (VO), and 3D object detection. The data for the visual odometry benchmark consists of 22  
104 image sequences, of which 11 are linked to ground-truth, and the others are primarily raw  
105 sensor data. The data includes eight classes: ‘Van’, ‘Car’, ‘Truck’, ‘Cyclist’, ‘Pedestrian’,

106 ‘Person (sitting)’, ‘Tram’ and ‘Misc’. The ground-truth for VO is the output of GPS/IMU  
107 localization. The trajectories are provided with the dataset. In (Geiger et al. 2012), the  
108 dataset was evaluated for stereo and optical flow using the average number of erroneous  
109 pixels, which is reported to be three pixels in terms of disparity and end-point, taking into  
110 account the calibration and laser measurement errors. The same data was also evaluated  
111 for 3D object detection and orientation estimation. The detection was performed iteratively  
112 using a bottom-top approach, starting from the largest overlap, measured by bounding box  
113 intersection over the union for assigning ground-truth labels (Geiger et al. 2012).

114 **SemanticKITTI** (Behley et al. 2019) is based on the KITTI dataset, mainly the se-  
115 quences provided for the OV task. The data can be used for various purposes; it provides  
116 dense point-wise annotation for 0–10 sequences, while 11–21 sequences are used for testing.  
117 The dataset is designed to perform three key tasks: semantic scene segmentation, semantic  
118 scene completion (i.e., the prediction of upcoming semantic scenes), and semantic scene seg-  
119 mentation of many sequential scans. Despite having a good variety of vehicle distribution,  
120 very few situations in semanticKITTI have more than eight people or four riders (Gao et al.  
121 2020). The ground classes, which total 24, include the most common classes—road, sidewalk,  
122 building, flora, and terrain—while motorcycle riders are uncommon but nonetheless present  
123 in more than 100k annotated sites. The labelling process relies on an off-the-shelf laser-based  
124 SLAM system (Behley and Stachniss 2018) to register and loop close the sequences.

125 **SemanticPOSS** (Pan et al. 2020) is another dataset that includes LiDAR scans with  
126 moving objects. The same data format as SemanticKITTI is employed. Similar to KITTI,  
127 point clouds for SemanticPOSS were collected using a moving vehicle outfitted with a Pan-  
128 dora module (HESAI 2022) and a GPS/IMU localization system. The Pandora combines  
129 LiDAR and cameras into a single module. The car travelled roughly 1.5 kilometres along a  
130 busy thoroughfare with cyclists and pedestrians present. Each point in the acquired data is  
131 labelled with a distinctive instance label for a dynamic item (car, people, rider). The data  
132 can be used to predict the accuracy/precision of 3D semantic segmentation and dynamic

133 objects and people (Gao et al. 2020; Pan et al. 2020). Compared to SemanticKITTI, SemanticPOSS has a smaller data size. Even though the resolution on horizontal LiDAR scans is  
134 higher, the spatial distribution of the LiDAR points is unbalanced (Gao et al. 2020).  
135

136 **SynthCity** (Griffiths and Boehm 2019) is an artificially generated dataset with labelled  
137 point clouds produced by full-colour mobile laser scanning. Each point is assigned one of  
138 the nine categorical labels: high vegetation, low vegetation, structures, scanning artefacts,  
139 vehicles, hardscape, man-made terrain, and natural landscape. The artificial point clouds  
140 are produced from urban and suburban landscapes that have been digitally modelled in  
141 the Blender 3D graphics application (Blender 2022). The primary purpose of the dataset’s  
142 release is semantic per-point classification, where each point has a local feature vector and  
143 a classification label. The dataset lacks instance IDs; hence, it is not suitable for instance  
144 segmentation.

145 A real-world dataset for gathering point clouds employing six cameras, five radars,  
146 and one LiDAR, each with a 360-degree field of view, is called **nuTonomy scenes**  
147 (**nuScenes**) (Caesar et al. 2020). The data is completely annotated with 3D bounding  
148 boxes, primarily for autonomous driving scenarios, and coupled with relevant map infor-  
149 mation. The data represents twenty-three classes, including road, pavement, ground, tree,  
150 building, pole-like, and others. NuScenes has a hundred times more photos and seven times  
151 more object annotations than the KITTI dataset. The dataset includes tracking annotation  
152 and can be used for tracking and detecting 3D objects (Chang et al. 2019).

153 The photo-realistic virtual world of the for-profit video game “Grand Theft Auto V” was  
154 the inspiration for the synthetic sequential point-cloud data known as the Grand Theft Auto  
155 V (**GTA5**) (Richter et al. 2016). The point cloud generating strategy relies on extracting  
156 a series of images from the game, using a pipeline to create the associated label, and then  
157 building a large-scale pixel-level semantic segmentation. Every 40<sup>th</sup> time frame is captured  
158 by RenderDoc. The game uses a variety of resource types, such as texture maps and geo-  
159 metric objects, to assemble scenes, making it easier to identify relationships between scene

160 constituents. Semantic annotations found in the KITTI dataset are three orders of magni-  
161 tude smaller than those found in the GTA5 dataset (Geiger et al. 2013; Richter et al. 2016).  
162 The 19 semantic classes presented in the dataset include e.g., road, building, sky, truck,  
163 person, traffic signal, and other road scene items. On two datasets, including KITTI (Geiger  
164 et al. 2013), where the training phase contained both real and synthetic data using minibatch  
165 stochastic gradient descent, the data was used to train semantic segmentation models and  
166 evaluated. The model trained with synthetic data created in GTA5 performs 2.6 better than  
167 the one trained without it.

168 The **Hilti SLAM 2021 challenge** dataset, which is a benchmark, collects various sensor  
169 modalities of mixed indoor-outdoor environments with varying lighting conditions; indoor  
170 sequences are for labs, offices, and construction sites, and outdoor sequences are for parking  
171 lots and construction sites (Helmberger et al. 2021). The data was collected using a portable  
172 platform with several sensors, including three IMUs (ADIS16445) with precise spatial and  
173 temporal calibration, two LiDARs (Ouster OS0-64 and Livox MID70), and five AlphaSense  
174 cameras (one stereo pair). This dataset’s primary goal is to encourage the creation of new  
175 SLAM algorithms that are resilient for demanding real-world settings like construction sites  
176 while still achieving high accuracy. A similar challenge was conducted in 2022; however,  
177 data was primarily collected for building sites and the Sheldonian Theatre in Oxford, UK,  
178 using a sensor suite installed on an aluminium platform and designed for handheld use. The  
179 suite includes a Hesai PandarXT-32 and a Sevensense Alphasense Core camera head with  
180 five 0.4MP global shutter cameras. Sensors are synchronized within 1 ms through Precision  
181 Time Protocol (PTP).

## 182 **METHODOLOGY**

183 This section introduces the configuration of our prototypical portable device as well  
184 as data collection and post-processing pipelines. In [Sensors and devices](#) subsection, the  
185 sensors constituting our handheld scanner and the computer used to process data streams.  
186 In addition, we describe the static scanner used by land surveyors to provide ground-truth



187 scans.

188 In [Intrinsic calibrations of the sensors](#) sub-section, we intrinsically calibrate both of the  
189 cameras comprising our prototypical scanner. If the cameras were not calibrated, then it  
190 would be rather impressive to accurately perform certain mathematical operations, such  
191 as the projection of LiDAR points onto the image spaces of the cameras. Each camera is  
192 calibrated separately (no stereo vision). The input to the intrinsic calibration is a stream of  
193 images portraying a calibration pattern of known dimensions from different points of view.  
194 The output is an intrinsic camera matrix and the corresponding lens distortion coefficients.

195 In [Extrinsic calibrations of the sensors](#) subsection, on the other hand, we estimate the  
196 rigid-body transformation between the LiDAR sensor and all the other sensors in PointPix.  
197 In the case of the LiDAR-RGB camera calibration, the scanner is put in a stationary position  
198 so that both sensors can view a special calibration target. The input to this calibration is  
199 streams of LiDAR points and the corresponding RGB images portraying the target from  
200 different points of view. The output is a rotation matrix and a translation vector describing  
201 the rigid-body transformation between the origins of these sensors. The LiDAR-NIR camera  
202 suite undergoes the same calibration procedure. When it comes to the LiDAR and the IMU,  
203 their calibration procedure is targetless. The input is a stream of LiDAR points and IMU  
204 messages recorded while walking slowly in an office. The output is a rotation matrix between  
205 the origins of the sensors.

206 The input to the [Data collection system of the hand-held scanner](#) sub-section is all the  
207 data streams coming from the four PointPix sensors during scanning on the construction site.  
208 The output is a recording of these streams as a .bag file. Since we scanned the construction  
209 site five times (once every month), there are five .bag files.

210 The [Ground-truth scans](#) sub-section describes a standard multi-view registration process  
211 in which the input is single-location TLS scans collected by land surveyors, and the output  
212 is a ground-truth scan. As in the previous paragraph, there are five ground-truth scans since  
213 we scanned the construction site five times.

214 Finally, [Ground-truth trajectories](#) sub-section describes how we recover the five ground-  
215 truth trajectories of our prototypical scanner. The input to this process is sequences of  
216 the LiDAR and IMU messages along with the ground-truth scans described in the previous  
217 paragraph. The output is the ground-truth trajectories of PointPix.

## 218 **Sensors and devices**

219 Figure 1a shows sensors we used while scanning on the construction site. There is a  
220 LiDAR (Velodyne VLP-16) at the top of our prototypical scanner and an RGB camera  
221 (Alvium U-319c, 3.2 MP) located directly below it. A NIR camera (Alvium 1800 U-501, 5.0  
222 MP) is placed to the left of the RGB camera, while an IMU (Xsens MTi-610) is fixed on  
223 the right. There is a handle at the bottom of the scanner, and all the mentioned sensors  
224 are fixed firmly to a specially constructed aluminium frame. The data was recorded and  
225 pre-processed on a laptop (MacBook Pro 2021, with emulated Ubuntu 20.04), to which all  
226 the sensors were connected.

227 When it comes to the ground-truth scans, land surveyors utilized a static laser scanner  
228 (a Leica RTC 360 shown in Figure 1b) to get exact scans that are then stitched together  
229 using proprietary software.

## 230 **Intrinsic calibrations of the sensors**

231 The Brown-Conardy model ([Brown 1966](#)) (otherwise known as the pinhole camera) is  
232 used to calibrate both cameras intrinsically. This calibration estimates intrinsic character-  
233 istics of a camera such as focal length and image center (encoded in a  $3 \times 3$  camera matrix  
234  $K$ ) as well as lens' distortion coefficients  $(k_1, k_2, k_3, k_4, k_5)$ . The procedure for estimating  
235 all these parameters is as follows: (1) several images portraying a calibration target of a  
236 particular pattern and known dimensions (for example, a checkerboard) need to be taken  
237 from different points of view; (2) a 3D model of the calibration pattern is then matched to  
238 these images, and as a result, a number of perspective projection matrices are computed  
239 through the means of constrained non-linear minimisation; (3) these projection matrices are  
240 then decomposed into the camera matrix among other matrices. The intrinsic matrices and

lens distortion coefficients for our RGB and NIR cameras are provided with the dataset.

The intrinsic parameters of the LiDAR have default values as described in the manufacturer’s documentation. These parameters describe a trigonometric function transforming LiDAR points encoded in the polar coordinate system to the Cartesian coordinate system. However, the users of our dataset do not need to use these parameters since our dataset stores LiDAR points only in the Cartesian system. Additionally, we use the Velodyne driver (O’Quin et al. 2010) and limit the LiDAR’s range to 60 metres.

### Extrinsic calibrations of the sensors

Extrinsic calibration of a pair of sensors aims at estimating the rigid-body transformation between the origins of these sensors. This transformation consists of a  $3 \times 3$  rotation matrix and a 3-element translation vector. Our LiDAR is extrinsically calibrated in a pair with all the other sensors comprising our prototypical scanner. The respective matrices of these pairs are stored along with the dataset.

In the case of a camera-LiDAR suite, we used the latest method by Beltrán et al. (Beltrán et al. 2022), which requires that both sensors observe a special calibration target from a couple of points of view. The rectangular target of specific dimensions has four ArUco markers (Garrido-Jurado et al. 2014) at the corners, and four circles cut out around the middle of the target. The calibrated camera detects the ArUco markers and computes the relative rigid-body transformation between the target and the camera. In parallel, a 3D model of the target is also fitted into streams of LiDAR points, which results in another rigid-body transformation between the target and the LiDAR. The difference between these two transformations results in the extrinsic calibration between the sensors.

Finally, there are various methods for a LiDAR-IMU calibration, which are relatively complex due to the fact that a LiDAR sensor suffers from motion distortion during movement and an IMU suffers from significant drift and a bias problem. These are known in the robotics community and we refer the reader to (Mishra et al. 2021; Lv et al. 2022) for more information on these topics. We used VINS-Mono (Nua and Jitianming 2020) for this calibration, as it is

268 one of the latest methods, and its procedure does not require a calibration target. Instead,  
269 the trajectory of both sensors is separately and iteratively estimated while slowly walking  
270 with the scanner nearby a feature rich place. A good example of such as a place is a corner  
271 of a room. It has easily detectable edges in LiDAR scans between two perpendicular walls  
272 and between a wall and a floor. Also planar features can be readily detect thanks to the  
273 presence of the walls, floors and ceilings) The difference between the trajectory of the two  
274 sensors yields the extrinsic calibration.

275 The configuration of the sensor frames can be seen in Figure 1 on the left. The  $x$ -axis  
276 of the LiDAR, RGB camera, and NIR camera face forwards when the scanner is held in its  
277 operational position, but the  $x$ -axis of the IMU points backwards. LiDAR's and IMU's  $z$ -axes  
278 face upwards, while the RGB and NIR cameras'  $y$ -axes point downwards. The remaining  
279 axes can be further deduced from the figure.

## 280 Data collection system of the hand-held scanner

281 In our portable data collection system we use *Robot Operating System* (ROS) (Quigley  
282 et al. 2009) as a framework to handle the data streams produced by all four sensors. The  
283 data processing pipeline is shown in more depth in Figure 2.

284 In the first major step (signal decoding, i.e., the top layer of Figure 2), all the sensors  
285 send their signals to the respective drivers running on a laptop. The NIR and RGB cameras  
286 are set to send signals to the Alviium drivers at 70 Hz and 60 Hz respectively. These are  
287 the maximal frequencies we could set the cameras for. The drivers decode the signal and  
288 publish NIR and RGB images to our ROS-based system at the same frequencies. The type  
289 of the outputted messages is `sensor_msgs/Image`.

290 Similarly, the LiDAR's raw signal is in the form of packets of the User Datagram Protocol  
291 (UDP), sent to the Velodyne driver at around 10 Hz. Each packet is decoded in the Velodyne  
292 driver as described in the Velodyne's manual. The output is a sequence of LiDAR points  
293 of type `sensor_msgs/PointCloud2` published to the ROS system at 10 Hz. Each LiDAR  
294 message contains maximally around 30 000 points. Each point consists of  $x$ -,  $y$ - and  $z$ -

295 coordinate as well as intensity information measuring the return strength of the reflected  
296 laser beam.

297 In a similar vein, the IMU sensor sends an encoded signal to its driver, which is then de-  
298 coded into `sensor_msgs/Imu` messages published to ROS at 400 Hz. Each message contains  
299 nine pieces of information describing the state of the IMU at a given time: three rotational  
300 velocities (around  $x$ -,  $y$ - and  $z$ -axes), three accelerations (in  $x$ -,  $y$ - and  $z$ -directions) and  
301 three-element vector characterizing the strength and direction of the magnetic field with  
302 respect to the current pose of the IMU.

303 Moving to the next step (in the middle of Figure 2), there are different approaches to  
304 synchronize sensors in time. Papers like (Faizullin et al. 2021) show that synchronization  
305 on a hardware level can be very accurate (less than 1 ms). On the flip side, though, this  
306 approach requires a custom design of a synchronization system, often including an additional  
307 microcontroller and bespoke wiring between sensors. Another approach would be to use the  
308 Precision Time Protocol (PTP) to synchronize the sensors, which would add additional  
309 Ethernet switches to our data collection system. However, most sensors do not have PTP  
310 support, and neither do ours.

311 Therefore, we decided to synchronize our LiDAR and cameras using a ROS synchroniza-  
312 tion policy (Open Robotics 2010) based on matching messages with timestamp differences  
313 smaller than ten milliseconds. We admit, however, that such time synchronization might not  
314 be ideal. Additionally, the high resolution images produced by our cameras take a couple  
315 of milliseconds to transport from the sensors to the laptop as the cables connecting them  
316 are around 1.5 meters long. We tried to estimate this lag experimentally, but it seems it is  
317 not constant. Therefore, the overall time synchronization error introduced is slightly higher  
318 than 10 ms. This means that if LiDAR points were projected onto a corresponding image  
319 (see [Practical application: Projecting LiDAR points onto corresponding images](#)), the mis-  
320 alignment between the projected points and the corresponding pixels in the image can reach  
321 around 20 pixels when our handheld scanner undergoes fast rotational movements. We hy-

322 pothesize that this misalignment might cause small inaccuracies in the trajectory estimation  
323 of SLAM algorithms that simultaneously use streams of images and LiDAR points for state  
324 estimation. However, the best algorithms do not do that as (Helmberger et al. 2021) show  
325 in their ranking.

326 In addition, due to issues with our NIR camera, we have divided the time synchronization  
327 process into two. If any of the three sensors temporarily ceased to function while the syn-  
328 chronization matched messages on their topics, the synchronization of all three topics would  
329 also halt. Sometimes, our NIR camera stops publishing images to our system, which would  
330 effectively stop the synchronization process. Therefore, we decided to synchronize RGB im-  
331 ages and LiDAR scans separately and publish them on the `/pp_rgb/synced2points` and  
332 `/pp_points/synced2rgb` topics, whereas NIR images are synchronized with already syn-  
333 chronized LiDAR scans. With this approach, we can continue to record synchronized RGB  
334 images, and LiDAR scans even if the NIR camera briefly malfunctions. Time synchronized  
335 LiDAR, RGB and NIR messages are the output of this step.

336 In the last part (recording, at the bottom of Figure 2), all data streams are recorded into  
337 a standard bag file. We additionally keep an eye on the three synchronized topics and the  
338 IMU messages (`/imu/data`) while they are being recorded to ensure that our data collection  
339 system works. We chose to capture IMU messages at 400 Hz since SLAM algorithms perform  
340 better when the IMU rate is higher (Shan et al. 2020).

## 341 **Ground-truth scans**

342 This section discusses the ground-truth scans and focuses on the post-processing pipeline  
343 used for their creation and refinement. The ground-truth data set contains five scans referred  
344 to as  $GT_i, i = 1, \dots, 5$ , collected over a period of four months on an active construction site,  
345 as explained in Section **DATASET: ConSLAM**.

346 We face two main issues during the data registration process of individual TLS scans:  
347 varying overlap and geometric discrepancies. Such issues make a 3D point cloud registration  
348 difficult and require manual adjustments to ensure high precision of the data alignment.

349 Such registration can take up to a few dozen minutes to be performed by an experienced  
 350 person.

351 Nevertheless, we need to be able to provide a registration error metric for the datasets  
 352 hampered by the issues mentioned above. There exist several error metrics which can be  
 353 used for this purpose. Judging by the commonly used point cloud-related software (Cloud  
 354 Compare, Point Cloud Library (PCL) and open3D) and recent papers (Huang et al. 2021;  
 355 Shu et al. 2021; Sun et al. 2022) Root Mean Square Error is the most commonly used  
 356 metric for measuring the fit of registered point clouds. We opted, therefore, for the distance-  
 357 constrained Root Mean Square Error (RMSE<sub>d</sub> for short), as provided in Definition 1.

358 **Definition 1** (RMSE<sub>d</sub>). *Let  $P_{data} \subset \mathbb{R}^3$  and  $P_{target} \subset \mathbb{R}^3$  be two point sets, and  $\gamma : P_{target} \rightarrow$   
 359  $P_{data}$  be the nearest-neighbour function. Then,*

$$360 \quad RMSE_d = \sqrt{\sum_{q \in S_d} \frac{\|\gamma(q) - q\|^2}{|S_d|}}, \quad (1)$$

361 where  $S_d = \{q \mid \|\gamma(q) - q\| < d\} \subset P_{target}$ .

362 The distance parameter  $d$  reduces the number of potential outliers in finding the correspond-  
 363 ing points between individual point clouds. Such outliers exist because the individual scans  
 364 may have low density in overlapping regions or are subject to noise caused by dynamic objects  
 365 or reflective surfaces. After a couple of trials, we have set the threshold to one centimetre.  
 366 As we can see in Table 2 the maximal error never reaches that threshold, indicating that  
 367 such disparity between the overlapping sets of points does not commonly occur.

368 In the next step, we discuss how the individual scans are stitched to each other. Each of  
 369 the  $GT_i$  sets, was obtained from a multi-view registration of  $M_i$  scans. Let  $\mathbb{P} = \{P_k \subset \mathbb{R}^3 \mid$   
 370  $1 \leq k \leq M\}$  denote a set of  $M$  point clouds, and let  $H_M$  denote a square binary matrix,  
 371 which encodes the registration relation of the elements of  $\mathbb{P}$ . More specifically,  $H_M(i, j) = 1$   
 372 if  $|P_i \cap P_j| = N \gg 0$ , and  $H_M(i, j) = 0$  otherwise. Finally, let  $\mathbb{G} = \{g_k \mid 1 \leq k \leq M, g_k \in$   
 373  $SE(3)\}$  be a set of rigid transformations. The multi-view registration problem can then be

374 formulated as

$$375 \quad E(g_1, \dots, g_M) = \sum_{i=1}^M \sum_{j=1}^M H_M(i, j) \sum_{k=1}^{N_j} f_l(\|d(g_j(p_k^j), g_j(q_k^j))\|^2), \quad (2)$$

376 where  $\{p_k^j \rightarrow q_k^j\}$  are the  $N_j$  closest point correspondences from point clouds  $P_i, P_j$ , and  $f_l$  is  
377 a loss function. In other words, we want to minimize the alignment error by summing up the  
378 contributions for every pair of overlapping views. The solutions  $g_1, \dots, g_M = \operatorname{argmin}(E)$  are  
379 the rigid transformations that align the  $M$  clouds in the least squares sense. For more infor-  
380 mation, we refer the reader to a technical report authored by Adrian Haarbach ([Haarbach](#)  
381 [2015](#)).

382 Having registered these scans, we have downsampled them using distance-based down-  
383 sampling implemented in CloudCompare 2.12.2 with a threshold of five millimetres. Finally,  
384 we compute  $\operatorname{RMSE}_d$  distances between overlapping point sets and present the results in  
385 Table 2. We admit that stricter uncertainty analysis would need to be performed if the  
386 registration of ground-truth scans was to be more experimental. In our paper, however, data  
387 collection using a TLS along with subsequent registration followed a strict land surveying  
388 guidelines and best practice. This means that the TLS scanner used was designed and cali-  
389 brated for land surveying purposes, the maximal distance between the locations of the TLS  
390 was 10 m and many of the resulting scans were not only registered by bundle adjustment but  
391 also georeferenced to control points whose coordinates were determined using a very precise  
392 theodolite. Therefore, we believe that we have minimised the impact of uncertainties in the  
393 registration and the computation of  $\operatorname{RMSE}_d$ .

## 394 **Ground-truth trajectories**

395 By registering the LiDAR’s successive messages to the ground-truth scans, we are able to  
396 reconstruct the ground-truth trajectory of the LiDAR. To achieve this, we play each recorded  
397 bag file and save each LiDAR message as a separate LiDAR scan to the PLY file format.  
398 Additionally, we run a SLAM algorithm on each recorded bag and store the estimated key



399 poses as text files. The key poses are those with a distance of one metre in between or with  
400 a change in rotation of 35 degrees. Then, the text files and LiDAR scans are matched using  
401 the timestamps assigned during data collection.

402 Our version of the ICP algorithm is then executed on each pose-scan pair. For each  
403 LiDAR scan, its edges are extracted in the same manner as in A-LOAM (Tong and Shaozu  
404 2018), and the corresponding pose is used as a rough estimate for this fine registration. The  
405 ICP iterations start with a threshold of 0.3 metres to build correspondences to the nearest  
406 points, with each iteration decreasing this threshold by a factor of 0.85. The maximal number  
407 of iterations is 20. The algorithm then converges when the RMSE/fitness is less than one  
408 centimetre or when the total difference between two iterations is less than five millimetres.

409 The resulting six degrees of freedom (6-DOF) transformations are named after the as-  
410 sociated LiDAR scans and are saved as  $4 \times 4$  matrices. These matrices, then, indicate  
411 the geometric adjustments that the LiDAR scans must undergo to be in alignment with  
412 the ground-truth scans. The ground-truth trajectory is then the accumulation of all these  
413 matrices. The recovered positions of the LiDAR and photorealistic representations of the  
414 ground-truth data are shown in Figures 3 and 4.

415 To automate the registration procedure outlined above, we write a computer program.  
416 Nevertheless, some of the LiDAR scans have failed to register. This includes the following  
417 situations: (1) the drift of the SLAM algorithm is high enough that the estimated poses  
418 are too distant to find accurate correspondences between the extracted LiDAR features and  
419 the ground-truth scans; and (2) the trajectory of our scanner is not fully covered by the  
420 ground-truth scans from the land surveying team.

421 At the time of writing this paper, we have correctly registered around 98% of all the key  
422 LiDAR scans for sequences 2-5. The LiDAR scans that failed to register to the ground-truth  
423 scans, or those whose registered poses visually seem out of line with the neighbouring ones,  
424 are listed in our repository. Also, we did not manage to recover the trajectory for the first  
425 sequence because the recorded data is of lower frequency than in the four other sequences.

426 This is because of unforeseen hardware-related problems occurred during the first scanning.  
427 Therefore, our SLAM algorithms fail when running a bag file from sequence 1.

### 428 **Anonymization of images**

429 We also anonymize faces appearing in the recorded RGB and NIR images by applying  
430 a strong Gaussian blur. This is done by a python package *deface* (Drawitsch 2021), which  
431 detects and blurs faces automatically. We also visually control each image and blur faces  
432 using manual tools in case *deface* fails (around 0.8% of all images).

### 433 **DATASET: ConSLAM**

434 We collected this dataset at Whiteley’s in London, one of the city’s premier department  
435 shops. It was first designed by John Belcher and John James Joass in 1911, and now the  
436 structure is undergoing redevelopment which includes tearing down the present shopping  
437 centre hidden beneath a preserved historic façade. The new development is mainly for  
438 constructing a brand new, six- to nine-story building and luxury retail, leisure, and residential  
439 spaces.

### 440 **Dataset structure**

441 Our ConSLAM dataset is available at <https://github.com/mac137/ConSLAM>. Table 3  
442 presents dates when the data were captured, along with the duration of the scans and the  
443 approximate length of their trajectories. Our dataset is organized as depicted in Figure 5.  
444 The directory contains ZIP files with files and folders from individual scans. The scans are  
445 numbered from 1 to 5, with 1 being the oldest and 5 denoting the most recent one. We  
446 encoded this fact with `data_unpacked_x.zip` where  $x = 1, \dots, 5$  in Figure 5.

447 The bag recordings (`recording.bag`) can be *played* using rosbag (Field et al. 2010).  
448 Each contains four topics with the stream of RGB and NIR images, LiDAR points and IMU  
449 messages. We have unpacked these data points and stored them in `lidar/`, `rgb/` and `nir/`  
450 folders. These data points are named after the timestamps of the sequential LiDAR scans  
451 recorded during scanning. The ground-truth point clouds created by land surveyors and are

452 stored in respective `groundtruth_scan.ply` files. Described in [Ground-truth trajectories](#)  
 453 subsection, the ground-truth key poses are stored in the last folder – `poses/`. The collection  
 454 of these poses makes up the ground-truth trajectory of the LiDAR sensor.

455 Calibration parameters are stored in `data_calib.zip` file. They include RGB and NIR  
 456 camera calibration matrices along with their distortion coefficients. Additionally, extrinsic  
 457 calibration matrices are stored there as well. While the LiDAR-RGB and LiDAR-NIR camera  
 458 pairs have full rigid-body transformation matrices estimated, only a rotation matrix between  
 459 the LiDAR and IMU is stored in our dataset.

### 460 **Practical application: Projecting LiDAR points onto corresponding images**

461 As an example, we take an extrinsic LiDAR-camera calibration matrix  $\mathbf{T}_{\text{RGB}}^{\text{LiDAR}}$  stored in  
 462 `calib_lidar2rgb.txt` and RGB intrinsic camera matrix for distorted images  $\mathbf{K}_{\text{dist}}^{\text{RGB}}$  along  
 463 with five distortion coefficients  $(k_1, k_2, k_3, k_4, k_5)$  from `calib_rgb.txt`. We define

$$464 \quad \mathbf{T}_{\text{RGB}}^{\text{LiDAR}} = \begin{bmatrix} \mathbf{R}_{\text{RGB}}^{\text{LiDAR}} & \mathbf{T}_{\text{RGB}}^{\text{LiDAR}} \\ \mathbf{0}_{1 \times 3} & 1 \end{bmatrix}, \quad (3)$$

465 where  $\mathbf{R}_{\text{RGB}}^{\text{LiDAR}} \in SO(3)$  is a  $3 \times 3$  rotation matrix from the LiDAR to the camera and  $\mathbf{T}_{\text{RGB}}^{\text{LiDAR}}$   
 466 is a  $3 \times 1$  translation vector also from the LiDAR to the camera.

467 Now, let us take an RGB image from the `rgb/` folder of any sequence from one to four,  
 468 undistort it and compute the RGB camera intrinsic matrix for undistorted images  $\mathbf{K}_{\text{undist}}^{\text{RGB}}$   
 469 using OpenCV package ([Bradski 2000](#)). We find the corresponding LiDAR scan in the  
 470 `lidar/` folder using image’s file name, and we iterate over the points. In order to project a  
 471 single LiDAR point  $\mathbf{x}_i = [x_i, y_i, z_i]^\top$  onto the undistorted image, we follow

$$472 \quad \begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = \mathbf{K}_{\text{undist}}^{\text{RGB}} \begin{bmatrix} \mathbb{K}_{3 \times 3} \\ \mathbf{0}_{1 \times 3} \end{bmatrix}^\top \mathbf{T}_{\text{RGB}}^{\text{LiDAR}-1} \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix}, \quad (4)$$

473 and then the pixel coordinates  $[u, v]^\top$  are recovered from the homogeneous coordinates as  
474 follows

$$475 \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} u'/w' \\ v'/w' \end{bmatrix}. \quad (5)$$

476 We refer the reader to Figure 6, which shows an example of LiDAR points projected onto  
477 the corresponding image.

### 478 **Practical application: evaluation of odometry/SLAM trajectories**

479 Here, we present how to compare the trajectory by odometry and SLAM algorithms  
480 against the ground-truth trajectory of our dataset. For this, we adjusted some of the code  
481 in a popular trajectory evaluation package – EVO (Grupp 2017). Further instructions on  
482 the details of this integration can be found in the ConSLAM’s repository.

483 Here, however, we describe how we evaluated one of the latest SLAM systems – LIO-SAM  
484 (Shan et al. 2020) as well as a public implementation of the LOAM algorithm – A-LOAM.  
485 We configure LIO-SAM and A-LOAM so that they can access our `pp_points/synced2rgb`  
486 and `imu/data` topics. We then launch these algorithms and run the bag file with sequence  
487 number five. We save the key poses by LIO-SAM and A-LOAM and organize them in the  
488 same way as the ground-truth poses in the `poses/` folder.

489 We also post-process the LIO-SAM and A-LOAM poses to ensure that their trajectories’  
490 first pose corresponds to the first pose in our ground-truth trajectory. For example, in the  
491 case of the sequence number five, it will be the pose whose name is `16595183259219555`.  
492 We also left-multiply all the LIO-SAM and A-LOAM poses by the inverse of their respective  
493 first poses so that their first poses are the identities and the other poses are relative to them.  
494 We then show the results of this comparison in Figure 7 together with the *Absolute Pose*  
495 *Error* (APE) evaluation in Figure 8.

496 It can be seen that there is a visible translational difference between the LIO-SAM and  
497 our ground-truth trajectory of around half a meter in the left part of the image. It can also  
498 be seen that the SLAM trajectory very closely follows our ground-truth in places where the

499 scanning started. These results make sense because the drift caused by such algorithms is  
500 small around the origin and grows the further the scanner is. On the other hand, A-LOAM  
501 has a larger drift, which resulted in a large deviation from our ground-truth trajectory. The  
502 above results validate the need for such datasets as ours and further research into SLAM.

## 503 **CONCLUSION AND FUTURE DIRECTION**

504 We introduced a new real-world dataset, the “ConSLAM”, recorded periodically by our  
505 hand-held scanner on a construction site. The primary motivation of the “ConSLAM dataset  
506 was to facilitate evaluating and comparing different odometry and SLAM algorithms in a  
507 construction setting. Our dataset is unique, as it brings sequences of data collected peri-  
508 odically on the same construction site, reflecting real-world construction use cases such as  
509 progress monitoring based on hand-held scanners or robots. Our dataset could serve as  
510 a testing battleground for further development of efficient algorithms, especially having in  
511 mind periodic use cases in construction.

512 As part of our future research, we aim to extend this dataset with semantic labels of  
513 individual building elements. We also aim to add features such as normal vectors in point  
514 clouds or occlusion boundaries in images for a better spatial understanding of the scene. We  
515 look forward to seeing how the research community will utilize our dataset.

## 516 **DATA AVAILABILITY STATEMENT**

517 The whole dataset described in this paper is accessible under the following link [https:](https://github.com/mac137/ConSLAM)  
518 [//github.com/mac137/ConSLAM](https://github.com/mac137/ConSLAM).

## 519 **ACKNOWLEDGEMENTS**

520 The authors would like to thank Laing O’Rourke for allowing access to their construc-  
521 tion site and collecting the ground-truth scans. We also acknowledge Romain Carriquiry-  
522 Borchiarri of Ubisoft France for his help in rendering some of the figures and Amanda Xu, a  
523 summer intern at the University of Cambridge, who helped us with anonymizing the images.  
524 This work is also supported by the EU Horizon 2020 BIM2TWIN: Optimal Construction

525 Management & Production Control project under agreement No. 958398. The first author  
526 would also like to thank BP, GeoSLAM, Laing O'Rourke, Topcon and Trimble for sponsoring  
527 his studentship funding.

## 528 REFERENCES

- 529 Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Stachniss, C., and Gall, J.  
530 (2019). "Semantickitti: A dataset for semantic scene understanding of lidar sequences."  
531 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 9297–9307.
- 532 Behley, J. and Stachniss, C. (2018). "Efficient surfel-based slam using 3d laser range data in  
533 urban environments.." *Robotics: Science and Systems*, Vol. 2018, 59.
- 534 Beltrán, J., Guindel, C., de la Escalera, A., and García, F. (2022). "Automatic extrinsic  
535 calibration method for lidar and camera sensor setups." *IEEE Transactions on Intelligent  
536 Transportation Systems*.
- 537 Blender (2022). "The freedom to create, <<https://www.blender.org>>. Accessed: 2022-10-05.
- 538 Bradski, G. (2000). "The OpenCV Library." *Dr. Dobb's Journal of Software Tools*.
- 539 Brown, D. (1966). "Decentering distortion of lenses." *Photogrammetric Engineering*, 444–462.
- 540 Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., Krishnan, A., Pan, Y.,  
541 Baldan, G., and Beijbom, O. (2020). "nusenes: A multimodal dataset for autonomous  
542 driving." *Proceedings of the IEEE/CVF conference on computer vision and pattern recog-  
543 nition*, 11621–11631.
- 544 Chang, M.-F., Lambert, J., Sangkloy, P., Singh, J., Bak, S., Hartnett, A., Wang, D., Carr,  
545 P., Lucey, S., Ramanan, D., et al. (2019). "Argoverse: 3d tracking and forecasting with  
546 rich maps." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern  
547 Recognition*, 8748–8757.
- 548 Drawitsch, M. (2021). "deface 1.1.1, <<https://pypi.org/project/deface/>>. Accessed: 2022-  
549 10-05.
- 550 Faizullin, M., Kornilova, A., and Ferrer, G. (2021). "Open-source lidar time synchronization  
551 system by mimicking gnss-clock.

552 Field, D., Leibs, J., Bowman, J., Thomas, D., and Perron, J. (2010). “Rosbag,  
553 <<https://wiki.ros.org/rosbag>>. Accessed: 2022-10-05.

554 Fritsch, J., Kuehnl, T., and Geiger, A. (2013). “A new performance measure and evalua-  
555 tion benchmark for road detection algorithms.” *16th International IEEE Conference on*  
556 *Intelligent Transportation Systems (ITSC 2013)*, IEEE, 1693–1700.

557 Gao, B., Pan, Y., Li, C., Geng, S., and Zhao, H. (2020). “Are we hungry for 3d li-  
558 dar data for semantic segmentation? a survey and experimental study.” *arXiv preprint*  
559 *arXiv:2006.04307*.

560 Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F. J., and Marín-Jiménez, M. J.  
561 (2014). “Automatic generation and detection of highly reliable fiducial markers under  
562 occlusion.” *Pattern Recognition*, 47(6), 2280–2292.

563 Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013). “Vision meets robotics: The kitti  
564 dataset.” *The International Journal of Robotics Research*, 32(11), 1231–1237.

565 Geiger, A., Lenz, P., and Urtasun, R. (2012). “Are we ready for autonomous driving? the kitti  
566 vision benchmark suite.” *2012 IEEE conference on computer vision and pattern recognition*,  
567 IEEE, 3354–3361.

568 Griffiths, D. and Boehm, J. (2019). “Synthcity: A large scale synthetic point cloud.” *arXiv*  
569 *preprint arXiv:1907.04758*.

570 Grupp, M. (2017). “evo: Python package for the evaluation of odometry and SLAM.,  
571 <<https://github.com/MichaelGrupp/evo>>.

572 Haarbach, A. (2015). “Multiview ICP, <<http://www.adrian-haarbach.de/mv-lm-icp/docs/mv-lm-icp.pdf>>.

574 Helmberger, M., Morin, K., Kumar, N., Wang, D., Yue, Y., Cioffi, G., and Scaramuzza, D.  
575 (2021). “The hilti slam challenge dataset.” *arXiv preprint arXiv:2109.11316*.

576 HESAI (2022). “Empower robotics, <[www.hesaitech.com](http://www.hesaitech.com)>. Accessed: 2022-10-05.

577 Huang, X., Mei, G., Zhang, J., and Abbas, R. (2021). “A comprehensive survey on point  
578 cloud registration.” *arXiv preprint arXiv:2103.02690*.

579 Khoshelham, K., Díaz, K., Vilariño, L., Peter, M., Kang, Z., and Acharya, D. (2017). “The  
580 isprs benchmark on indoor modelling.” *Int. Arch. Photogramm. Remote Sens. Spatial Inf.*  
581 *Sci., XLII-2/W7*, 367–372.

582 Lv, J., Zuo, X., Hu, K., Xu, J., Huang, G., and Liu, Y. (2022). “Observability-aware intrinsic  
583 and extrinsic calibration of lidar-imu systems.” *IEEE Transactions on Robotics*, 1–20.

584 Menze, M. and Geiger, A. (2015). “Object scene flow for autonomous vehicles.” *Proceedings*  
585 *of the IEEE conference on computer vision and pattern recognition*, 3061–3070.

586 Mishra, S., Pandey, G., and Saripalli, S. (2021). “Target-free extrinsic calibration of a 3d-lidar  
587 and an imu.” *2021 IEEE International Conference on Multisensor Fusion and Integration*  
588 *for Intelligent Systems (MFI)*, IEEE, 1–7.

589 Nua, C. and Jitianming, M. (2020). “Automatic calibration of 3d lidar and imu extrinsics,  
590 <[https://github.com/chennuo0125-HIT/lidar\\_imu\\_calib](https://github.com/chennuo0125-HIT/lidar_imu_calib)>. Accessed: 2022-10-05.

591 Open Robotics (2010). “Approximate time filters, <[https://wiki.ros.org/message\\_filters/ApproximateTime](https://wiki.ros.org/message_filters/ApproximateTime)>.  
592 Accessed: 2022-10-05.

593 O’Quin, J., Beeson, P., Quinlan, M., and Liu, Y. (2010). “Ros velodyne driver,  
594 <[http://wiki.ros.org/velodyne\\_driver](http://wiki.ros.org/velodyne_driver)>. Accessed: 2022-10-05.

595 Pan, Y., Gao, B., Mei, J., Geng, S., Li, C., and Zhao, H. (2020). “Semanticpos: A point  
596 cloud dataset with large quantity of dynamic instances.” *2020 IEEE Intelligent Vehicles*  
597 *Symposium (IV)*, IEEE, 687–693.

598 Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R.,  
599 and Ng, A. (2009). “Ros: an open-source robot operating system.” *Proc. of the IEEE Intl.*  
600 *Conf. on Robotics and Automation (ICRA) Workshop on Open Source Robotics*, Kobe,  
601 Japan (May).

602 Richter, S. R., Vineet, V., Roth, S., and Koltun, V. (2016). “Playing for data: Ground truth  
603 from computer games.” *European conference on computer vision*, Springer, 102–118.

604 Shan, T., Englot, B., Meyers, D., Wang, W., Ratti, C., and Daniela, R. (2020). “Lio-  
605 sam: Tightly-coupled lidar inertial odometry via smoothing and mapping.” *IEEE/RSJ*



606 *International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 5135–5142.

607 Shu, Z., Cao, S., Jiang, Q., Xu, Z., Tang, J., and Zhou, Q. (2021). “Pairwise registration  
608 algorithm for large-scale planar point cloud used in flatness measurement.” *Sensors*, 21(14),  
609 4860.

610 Sturm, J., Engelhard, N., Endres, F., Burgard, W., and Cremers, D. (2012). “A bench-  
611 mark for the evaluation of rgb-d slam systems.” *Proc. of the International Conference on*  
612 *Intelligent Robot Systems (IROS)* (Oct.).

613 Sun, Z., Zhang, R., Hu, J., and Liu, X. (2022). “Probability re-weighted 3d point cloud reg-  
614 istration for missing correspondences.” *Multimedia Tools and Applications*, 81(8), 11107–  
615 11126.

616 Tong, Q. and Shaozu, C. (2018). “Advanced implementation of loam,  
617 <<https://github.com/HKUST-Aerial-Robotics/A-LOAM>>. Accessed: 2022-10-05.

618 Trzeciak, M., Pluta, K., Fathy, Y., Alcalde, L., Chee, S., Bromley, A., Brilakis, I., and Alliez,  
619 P. (2022). “Conslam: Periodically collected real-world construction dataset for slam and  
620 progress monitoring.

621 **List of Tables**

622 1 Summary of existing sequential point-cloud datasets . . . . . 26

623 2 RMSE<sub>d</sub> distance for ground-truths dataset. The measurements are recorded

624 in centimetres . . . . . 27

625 3 Metadata of the collected sequences . . . . . 28

TABLE 1. Summary of existing sequential point-cloud datasets

Name	Real (R)/ Synthetic (S)	Indoor (I)/ Outdoor (O)	Trajectory	Sector	Applications	Sensors
KITTI (Geiger et al. 2012)	R	I	yes	urban and transport	autonomous vehicles (Menze and Geiger 2015; Fritsch et al. 2013), 3D object detection and visual odometry (Geiger et al. 2012)	4x colour and greyscale stereo cameras, a laser scanner (Velodyne), 4x Edmund optics lenses, GPS navigation systems
Semantic-KITTI (Behley et al. 2019)	R	O	yes	urban/road	semantic segmentation of a scene, semantic scene completion (i.e., predicting future semantic scenes), and semantic segmentation of multiple sequential scans (Behley et al. 2019)	relying on the data collected by laser scanner (Velodyne) in the KITTI dataset
Semantic-POSS (Pan et al. 2020)	R	O	no	urban/road	prediction accuracy of dynamic objects and people (Gao et al. 2020) and 3D semantic segmentation (Pan et al. 2020)	Pandora module (LiDAR, mono and colour cameras) and GPS/IMU
SynthCity (Griffiths and Boehm 2019)	S	O	yes	urban/suburban environments	point-cloud classification (Griffiths and Boehm 2019)	mobile laser scanning
GTA5 (Richter et al. 2016)	S	O	no	urban/road	semantic segmentation and scene understanding (Richter et al. 2016)	frames extracted from "Grand Theft Auto V" video game; from a car perspective
nuScenes (Caesar et al. 2020)	R	O	yes	urban/road and autonomous driving	object detection and tracking, segmentation (Caesar et al. 2020)	6 cameras, five radars and 1 LiDAR, all with full 360 degree field of view
HILTI- OXFORD (Helmberger et al. 2021)	R	I & O	yes	built environment	accuracy of odometry/SLAM algorithms in the built environment	5 AlphaSense grayscale cameras, 2 LiDARs (Ouster OS0-64 and Livox MID70), and 3 IMUs (ADIS16445)
ConSLAM	R	I	yes	construction	accuracy of odometry/SLAM algorithms for periodic use-cases in construction	LiDAR (Velodyne VLP-16), RGB camera (Alvium U-319c, 3.2 MP), a NIR camera (Alvium 1800 U-501, 5.0 MP) and an IMU (Xsens MTi-610)(see Figure 1)

**TABLE 2.**  $\text{RMSE}_d$  distance for ground-truths dataset. The measurements are recorded in centimetres

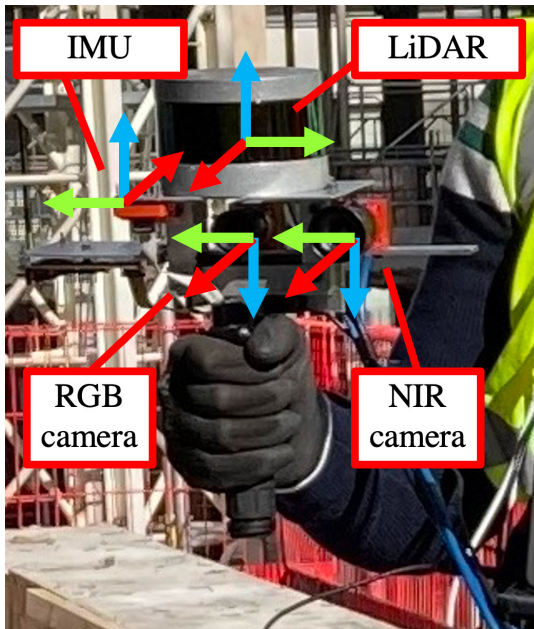
<b>Dataset Name</b>	$\approx \text{min}_{\text{RMSE}_d}$	$\approx \text{max}_{\text{RMSE}_d}$	$\approx \text{mean}_{\text{RMSE}_d}$
$GT_1$	0.319	0.950	0.676
$GT_2$	0.262	0.980	0.605
$GT_3$	0.327	0.983	0.607
$GT_4$	0.360	0.902	0.637
$GT_5$	0.345	0.994	0.665

**TABLE 3.** Metadata of the collected sequences

<b>Sequence</b>	<b>Date</b>	<b>Duration (s)</b>	<b>Approx. length (m)</b>
1	15/03/2022	436	235
2	26/04/2022	420	225
3	09/06/2022	630	340
4	29/06/2022	506	275
5	03/08/2022	589	320

626	<b>List of Figures</b>	
627	1	Data acquisition devices: our prototypical hand-held scanner (a) and a static
628		TLS scanner used to collect ground-truth scans (b). Red, green and blue
629		arrows in the left image represent $x$ -, $y$ - and $z$ -axes, respectively . . . . .
630		(a) PointPix hand-held scanner . . . . .
631		(b) Static TLS scanner . . . . .
632	2	Processing data streams on construction site . . . . .
633	3	Top-view visualization of the ground-truth datasets . . . . .
634		(a) Top view visualisation of $GT_1$ . . . . .
635		(b) Top view visualisation of $GT_2$ . . . . .
636		(c) Top view visualisation of $GT_3$ . . . . .
637		(d) Top view visualisation of $GT_4$ . . . . .
638		(e) Top view visualisation of $GT_5$ . . . . .
639	4	Close-up visualization of the ground-truth datasets. The images include also
640		the LiDAR positions depicted by red spheres. The positions have been con-
641		nected to provide approximated paths/trajectories . . . . .
642		(a) Close-up view of the construction site from $GT_1$ . . . . .
643		(b) Corridor view of the construction site from $GT_1$ . . . . .
644		(c) Close-up view of the construction site from $GT_2$ . . . . .
645		(d) Corridor view of the construction site from $GT_2$ . . . . .
646		(e) Close-up view of the construction site from $GT_3$ . . . . .
647		(f) Corridor view of the construction site from $GT_3$ . . . . .
648		(g) Close-up view of the construction site from $GT_4$ . . . . .
649		(h) Corridor view of the construction site from $GT_4$ . . . . .
650		(i) Close-up view of the construction site from $GT_5$ . . . . .
651		(j) Corridor view of the construction site from $GT_5$ . . . . .
652	5	Dataset folder structure . . . . .

653	6	Example of projecting of LiDAR points onto the corresponding image . . . . .	36
654	7	Visualization of the sequence five's the trajectories obtained from A-LOAM	
655		and LIO-SAM against our ground-truth trajectory . . . . .	37
656	(a)	Trajectories viewed in a function of time for $x$ -, $y$ - and $z$ -axis . . . . .	37
657	(b)	Trajectories viewed in a function of time for the <i>yaw</i> , <i>pitch</i> and <i>roll</i>	
658		rotations . . . . .	37
659	(c)	The $xy$ -plane view of the trajectories . . . . .	37
660	8	The plots showing the Absolute Pose Error (APE) for the LIO-SAM and	
661		A-LOAM trajectories . . . . .	38
662	(a)	Color-coded APE for LIO-SAM viewed in the $xy$ -plane . . . . .	38
663	(b)	Color-coded APE for A-LOAM viewed in the $xy$ -plane . . . . .	38
664	(c)	Plot of APE for LIO-SAM . . . . .	38
665	(d)	Plot of APE for A-LOAM . . . . .	38



(a) PointPix hand-held scanner



(b) Static TLS scanner

**Fig. 1.** Data acquisition devices: our prototypical hand-held scanner (a) and a static TLS scanner used to collect ground-truth scans (b). Red, green and blue arrows in the left image represent  $x$ -,  $y$ - and  $z$ -axes, respectively



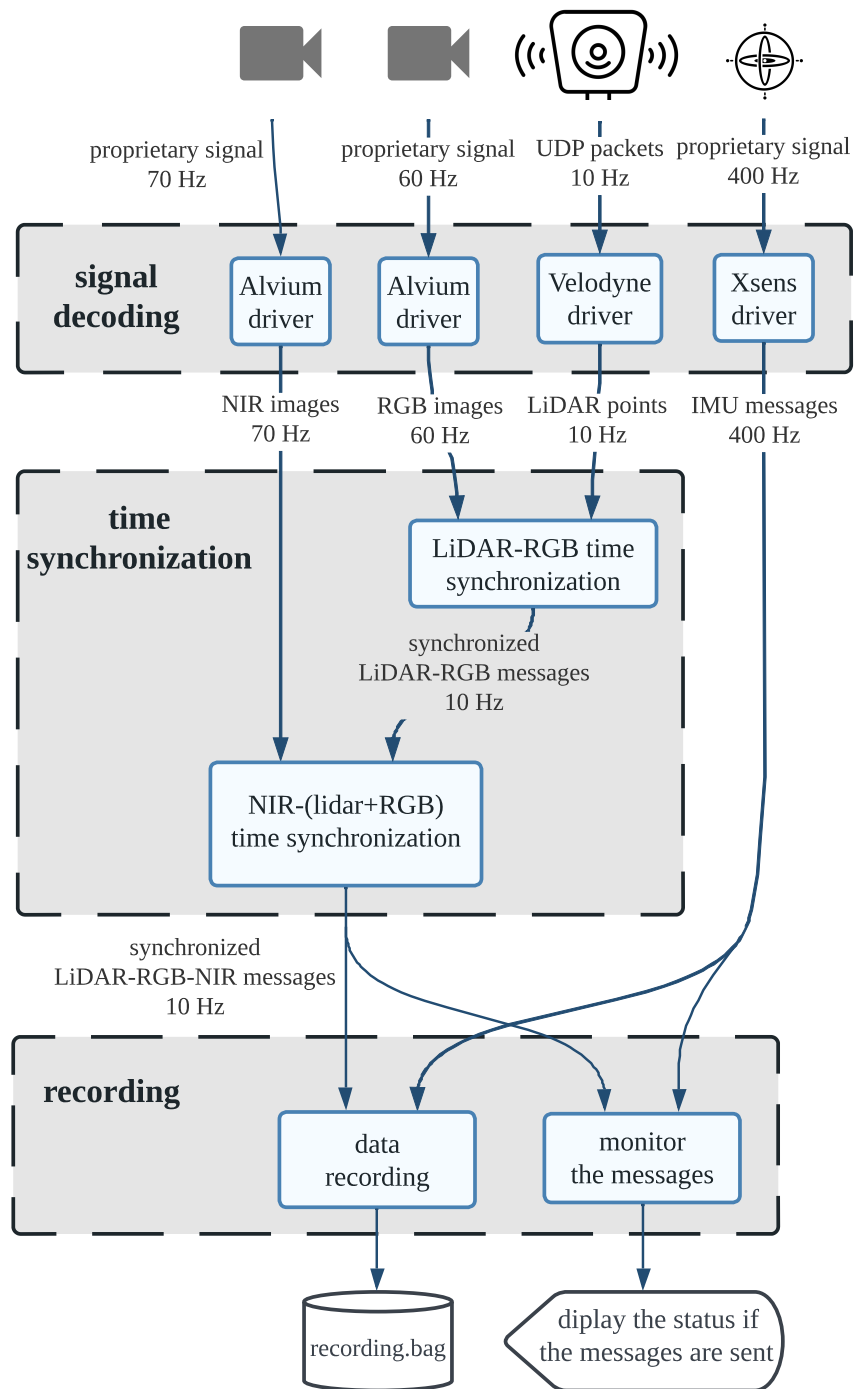
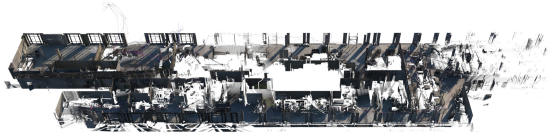


Fig. 2. Processing data streams on construction site



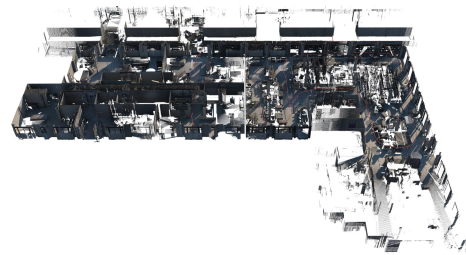
(a) Top view visualisation of  $GT_1$



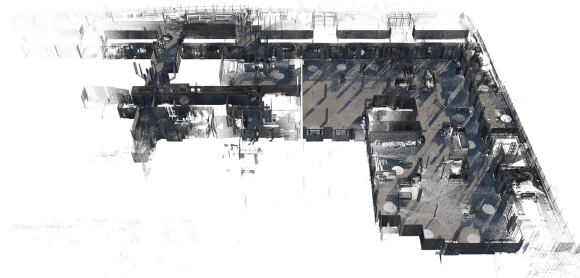
(b) Top view visualisation of  $GT_2$



(c) Top view visualisation of  $GT_3$

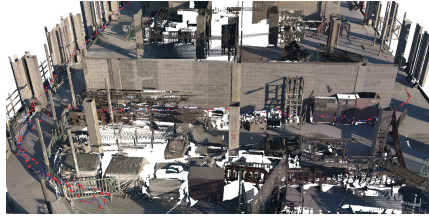


(d) Top view visualisation of  $GT_4$



(e) Top view visualisation of  $GT_5$

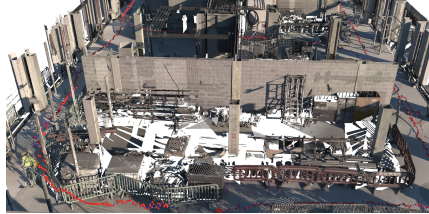
**Fig. 3.** Top-view visualization of the ground-truth datasets



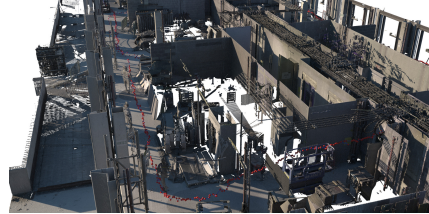
(a) Close-up view of the construction site from  $GT_1$



(b) Corridor view of the construction site from  $GT_1$



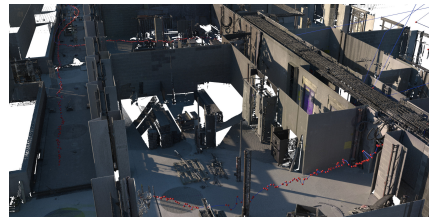
(c) Close-up view of the construction site from  $GT_2$



(d) Corridor view of the construction site from  $GT_2$



(e) Close-up view of the construction site from  $GT_3$



(f) Corridor view of the construction site from  $GT_3$



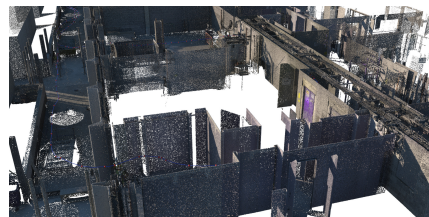
(g) Close-up view of the construction site from  $GT_4$



(h) Corridor view of the construction site from  $GT_4$

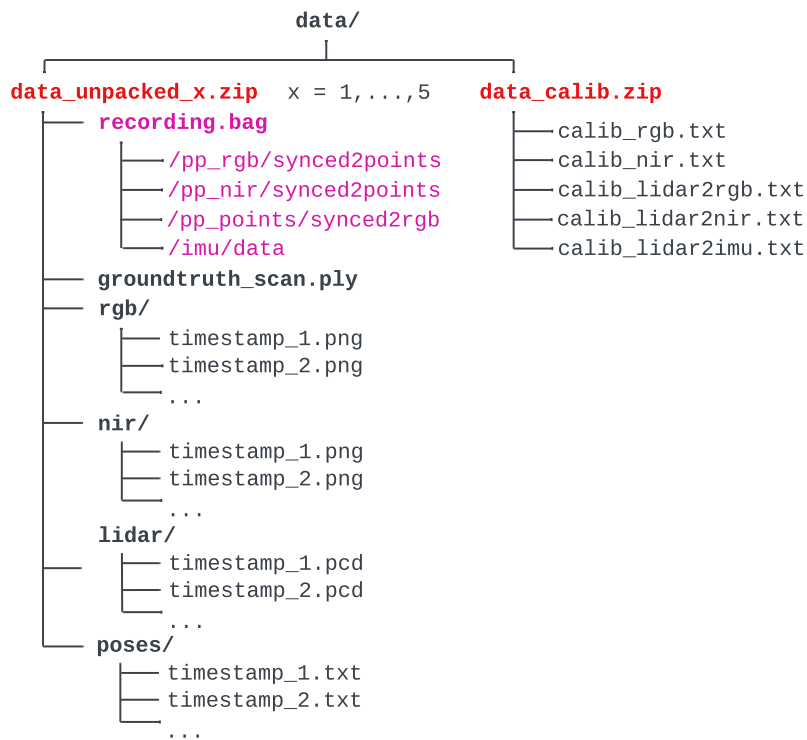


(i) Close-up view of the construction site from  $GT_5$



(j) Corridor view of the construction site from  $GT_5$

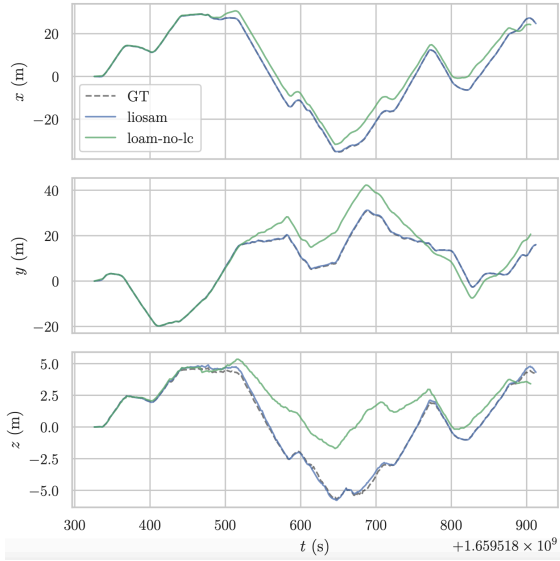
**Fig. 4.** Close-up visualization of the ground-truth datasets. The images include also the LiDAR positions depicted by red spheres. The positions have been connected to provide approximated paths/trajectories



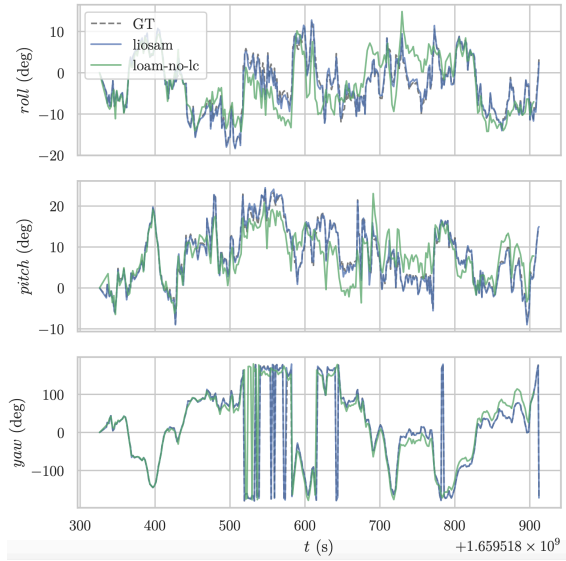
**Fig. 5.** Dataset folder structure



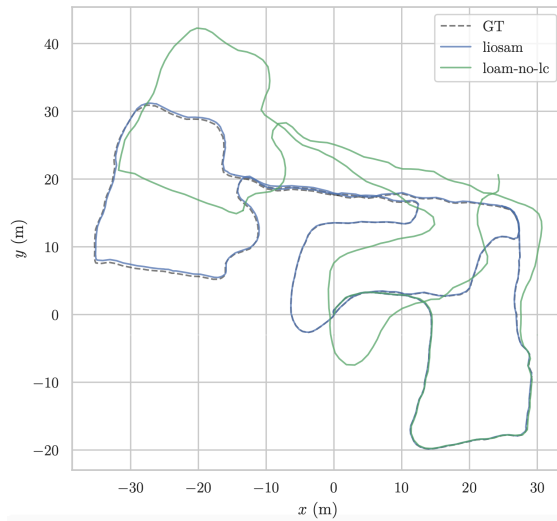
**Fig. 6.** Example of projecting of LiDAR points onto the corresponding image



(a) Trajectories viewed in a function of time for  $x$ -,  $y$ - and  $z$ -axis

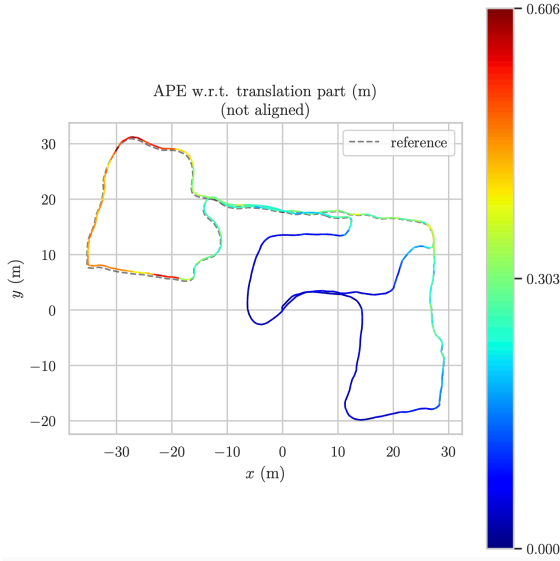


(b) Trajectories viewed in a function of time for the  $yaw$ ,  $pitch$  and  $roll$  rotations

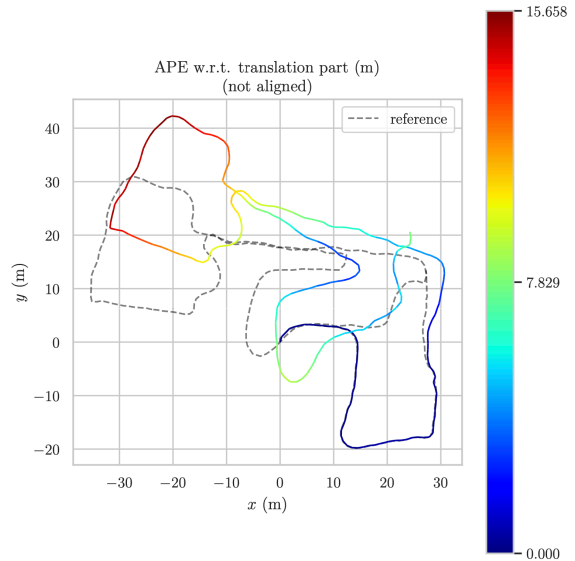


(c) The  $xy$ -plane view of the trajectories

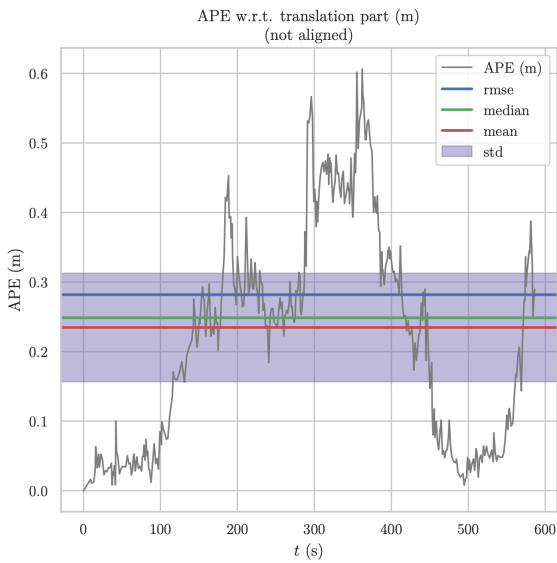
**Fig. 7.** Visualization of the sequence five's the trajectories obtained from A-LOAM and LIO-SAM against our ground-truth trajectory



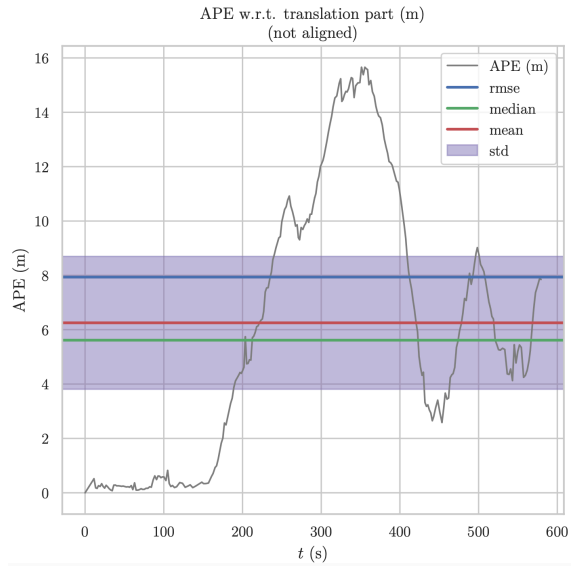
(a) Color-coded APE for LIO-SAM viewed in the  $xy$ -plane



(b) Color-coded APE for A-LOAM viewed in the  $xy$ -plane



(c) Plot of APE for LIO-SAM



(d) Plot of APE for A-LOAM

**Fig. 8.** The plots showing the Absolute Pose Error (APE) for the LIO-SAM and A-LOAM trajectories