ConSLAM: Construction Data Set for SLAM

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

ConSLAM: Construction Dataset for SLAM

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

[1]Department of Engineering, University of Cambridge, CB2 1PZ, UK, mpt35@cam.ac.uk

[2]Inria Sophia Antipolis-Méditerranée, 06902 Valbonne, France, kacper.pluta@inria.fr

[3]Department of Engineering, University of Cambridge, CB2 1PZ, UK, yafa2@cam.ac.uk

[4]Laing O'Rourke, DA2 6SN, Dartford, UK, lalcalde@laingorourke.com

[5]Laing O'Rourke, DA2 6SN, Dartford, UK, schee@laingorourke.com

[6]Laing O'Rourke, DA2 6SN, Dartford, UK, abromley@laingorourke.com

[7]Department of Engineering, University of Cambridge, CB2 1PZ, UK, ib340@cam.ac.uk

[8]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)

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

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

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

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

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

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

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

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

EXISTING DATASETS

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

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

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

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

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

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 higher, the spatial distribution of the LiDAR points is unbalanced (Gao et al. 2020).

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

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

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

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

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

METHODOLOGY

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

scans.

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

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

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

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

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

Sensors and devices

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

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

Intrinsic calibrations of the sensors

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

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

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

Data collection system of the hand-held scanner

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

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

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

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

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

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

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

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

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

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

Ground-truth scans

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

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

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

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

Definition 1 (RMSE_d)**.** *Let $P_{data} \subset \mathbb{R}^3$ and $P_{target} \subset \mathbb{R}^3$ be two point sets, and $\gamma : P_{target} \to P_{data}$ be the nearest-neighbour function. Then,*

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

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

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

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

formulated as

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

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 a loss function. In other words, we want to minimize the alignment error by summing up the contributions for every pair of overlapping views. The solutions $g_1, \ldots, g_M = argmin(E)$ are the rigid transformations that align the M clouds in the least squares sense. For more information, we refer the reader to a technical report authored by Adrian Haarbach (Haarbach 2015).

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

Ground-truth trajectories

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

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

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

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

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

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

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

Anonymization of images

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

DATASET: ConSLAM

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

Dataset structure

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

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

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

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

Practical application: Projecting LiDAR points onto corresponding images

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

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

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

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

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

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

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} u'/w' \\ v'/w' \end{bmatrix}. \tag{5}$$

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

Practical application: evaluation of odometry/SLAM trajectories

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

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

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

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

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

CONCLUSION AND FUTURE DIRECTION

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

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

DATA AVAILABILITY STATEMENT

The whole dataset described in this paper is accessible under the following link https://github.com/mac137/ConSLAM.

ACKNOWLEDGEMENTS

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

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

REFERENCES

Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Stachniss, C., and Gall, J. (2019). "Semantickitti: A dataset for semantic scene understanding of lidar sequences." *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 9297–9307.

Behley, J. and Stachniss, C. (2018). "Efficient surfel-based slam using 3d laser range data in urban environments.." *Robotics: Science and Systems*, Vol. 2018, 59.

Beltrán, J., Guindel, C., de la Escalera, A., and García, F. (2022). "Automatic extrinsic calibration method for lidar and camera sensor setups." *IEEE Transactions on Intelligent Transportation Systems*.

Blender (2022). "The freedom to create, <https://www.blender.org>. Accessed: 2022-10-05.

Bradski, G. (2000). "The OpenCV Library." *Dr. Dobb's Journal of Software Tools*.

Brown, D. (1966). "Decentering distortion of lenses." *Photogrammetric Engineering*, 444–462.

Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., and Beijbom, O. (2020). "nuscenes: A multimodal dataset for autonomous driving." *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 11621–11631.

Chang, M.-F., Lambert, J., Sangkloy, P., Singh, J., Bak, S., Hartnett, A., Wang, D., Carr, P., Lucey, S., Ramanan, D., et al. (2019). "Argoverse: 3d tracking and forecasting with rich maps." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8748–8757.

Drawitsch, M. (2021). "deface 1.1.1, <https://pypi.org/project/deface/>. Accessed: 2022-10-05.

Faizullin, M., Kornilova, A., and Ferrer, G. (2021). "Open-source lidar time synchronization 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-
573 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>. 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*.

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

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

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

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

Nua, C. and Jitianming, M. (2020). "Automatic calibration of 3d lidar and imu extrinsics, <https://github.com/chennuo0125-HIT/lidar_imu_calib>. Accessed: 2022-10-05.

Open Robotics (2010). "Approximate time filters, <https://wiki.ros.org/message_filters/ApproximateTime Accessed: 2022-10-05.

O'Quin, J., Beeson, P., Quinlan, M., and Liu, Y. (2010). "Ros velodyne driver, <http://wiki.ros.org/velodyne_driver>. Accessed: 2022-10-05.

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

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

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

Shan, T., Englot, B., Meyers, D., Wang, W., Ratti, C., and Daniela, R. (2020). "Lio-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.

List of Tables

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, legmentation (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)

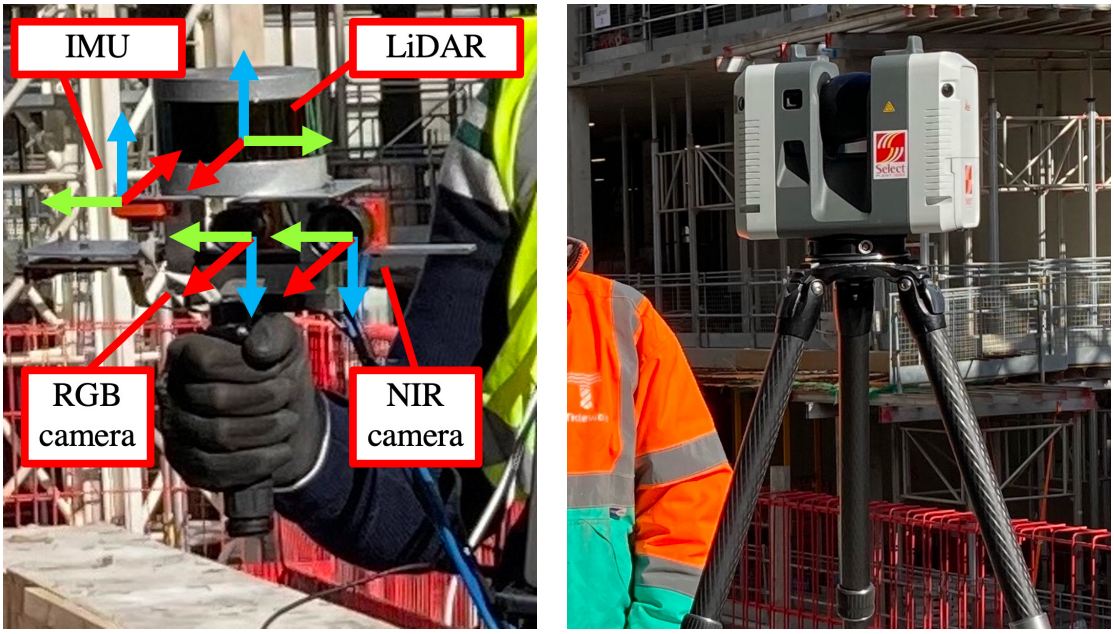
TABLE 2. RMSE_d distance for ground-truths dataset. The measurements are recorded in centimetres

Dataset Name	$\approx \mathbf{min}_{\mathbf{RMSE}_d}$	$\approx \mathbf{max}_{\mathbf{RMSE}_d}$	$\approx \mathbf{mean}_{\mathbf{RMSE}_d}$
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

Sequence	**Date**	**Duration (s)**	**Approx. length (m)**
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

List of Figures



(a) PointPix hend-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

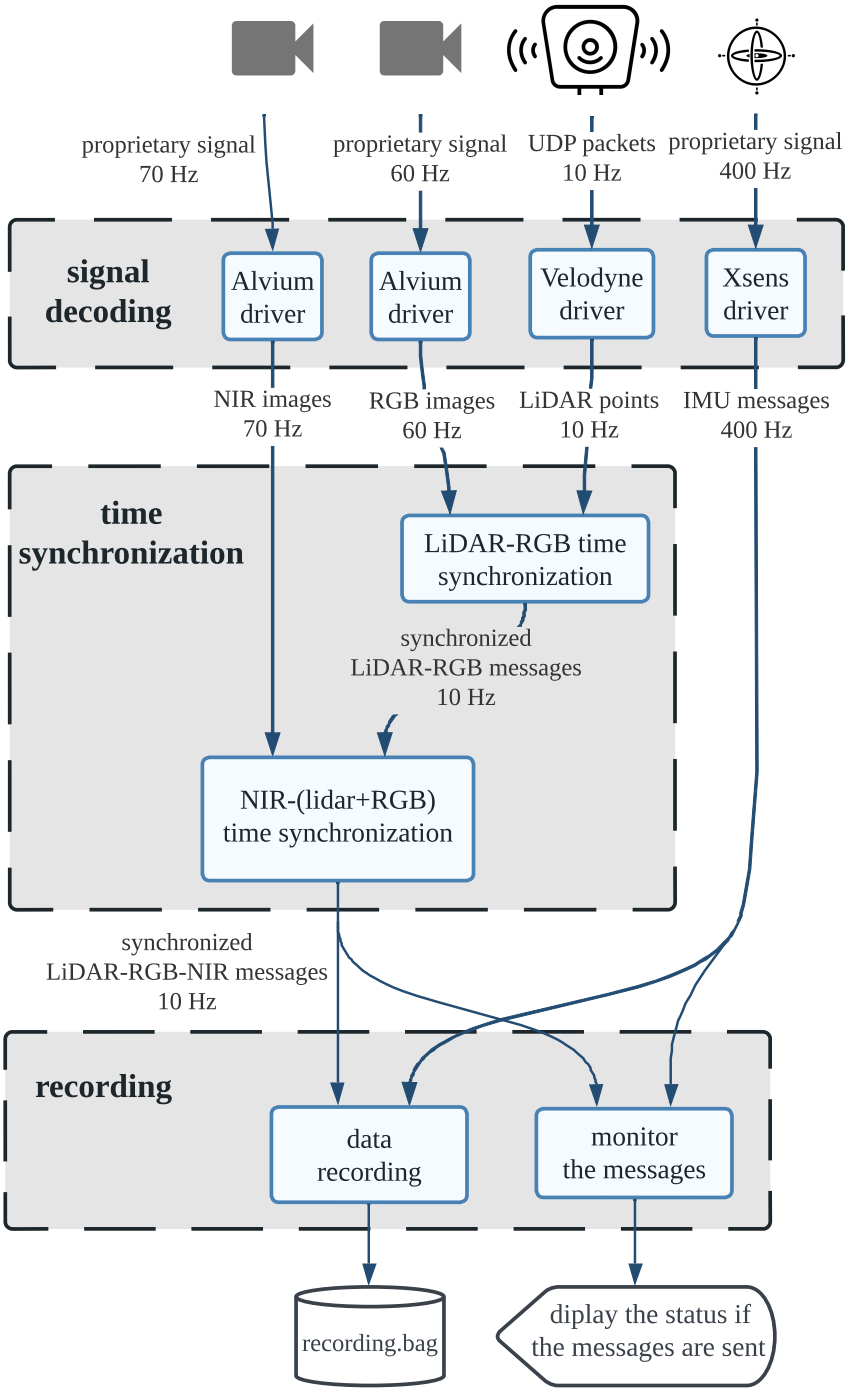
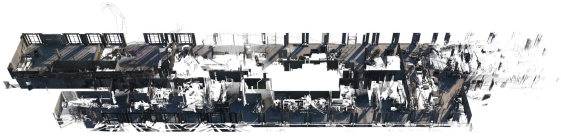
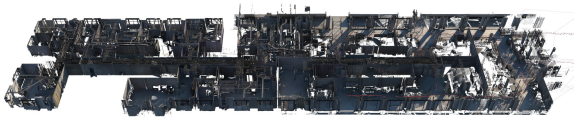


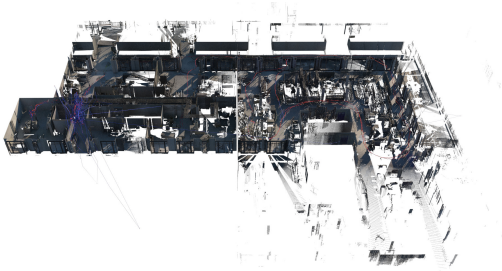
Fig. 2. Processing data streams on construction site



(a) Top view visualisation of GT_1

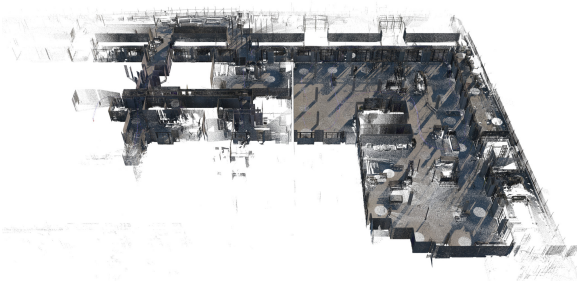


(b) Top view visualisation of GT_2



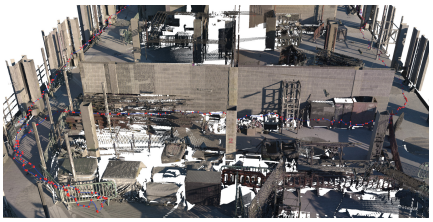
(c) Top view visualisation of GT_3

(d) Top view visualisation of GT_4

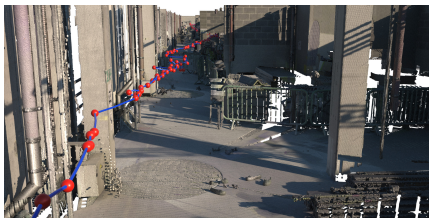


(e) Top view visualisation of GT_5

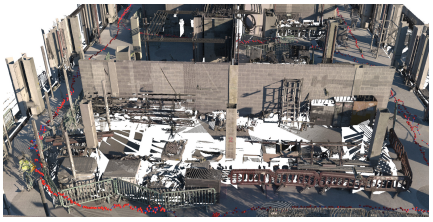
Fig. 3. Top-view visualization of the ground-truth datasets



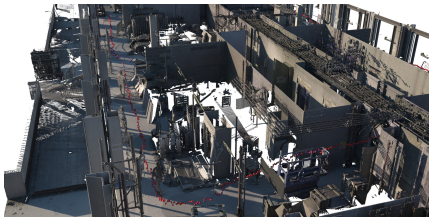
(a) Close-up view of the construction site from GT_1



(b) Corridor view of the construction site from GT_1

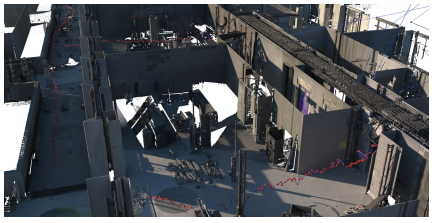


(c) Close-up view of the construction site from GT_2



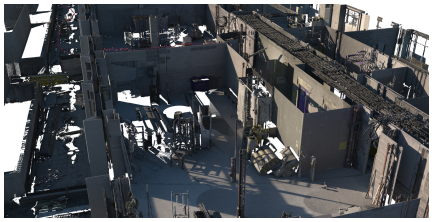
(d) Corridor view of the construction site from GT_2

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



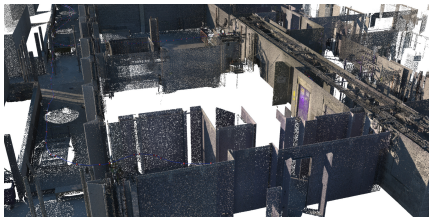
(f) Corridor view of the construction site from GT_3

(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

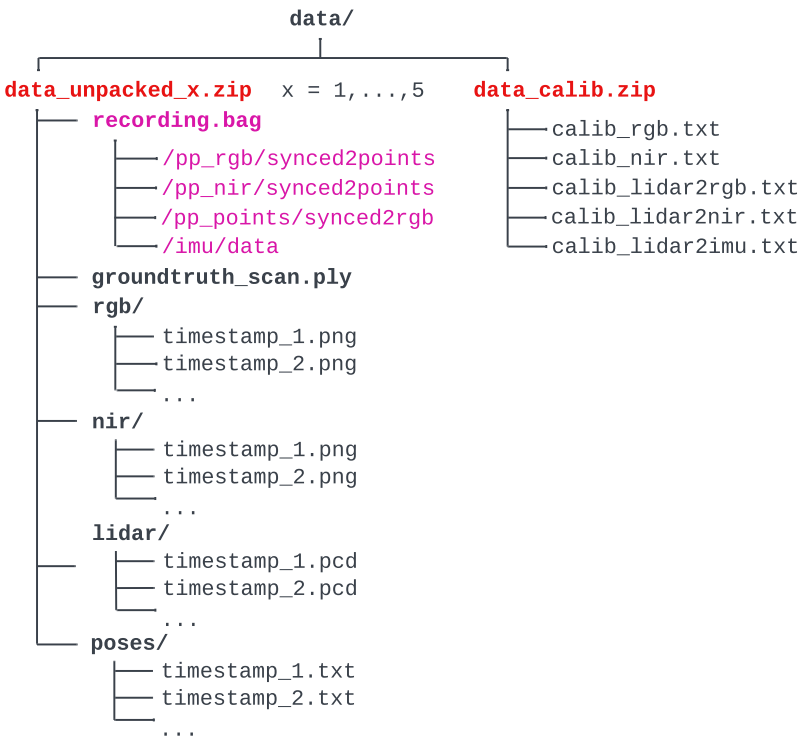
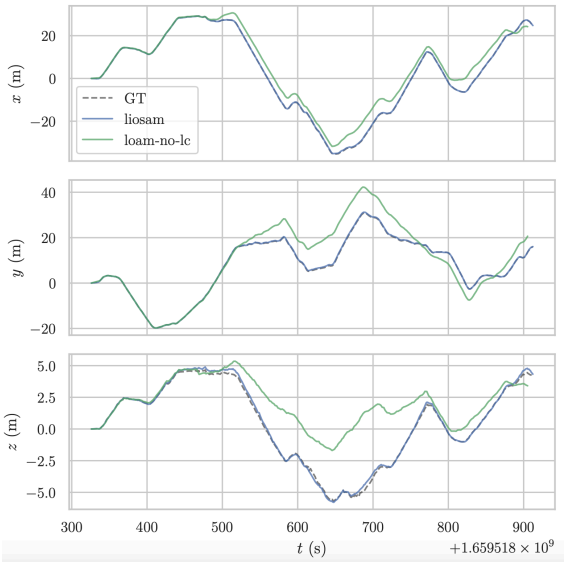
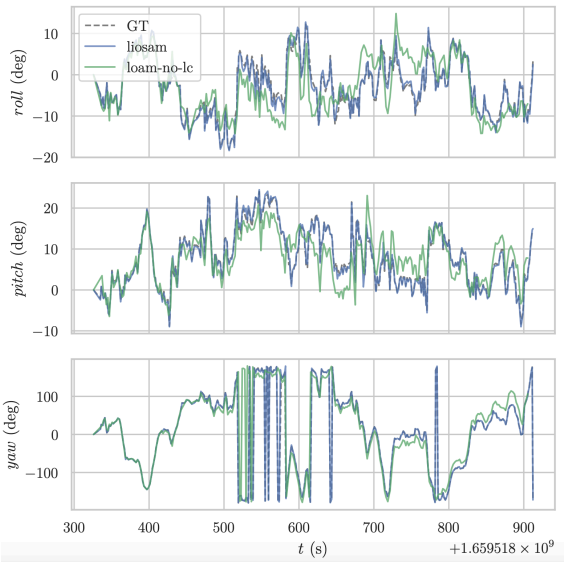


Fig. 5. Dataset folder structure

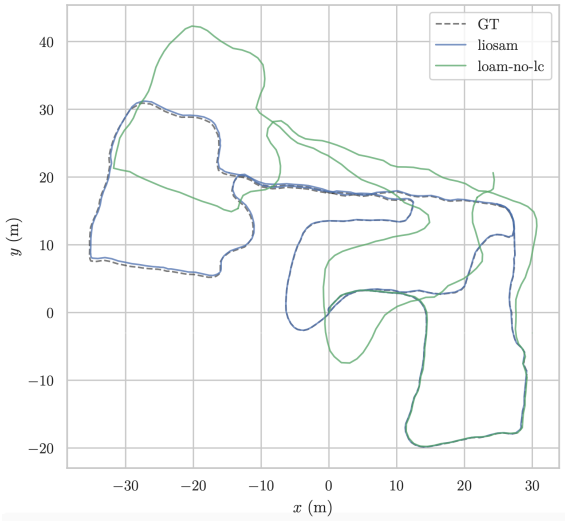
Fig. 6. Example of projecting of LiDAR points onto the corresponding image



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

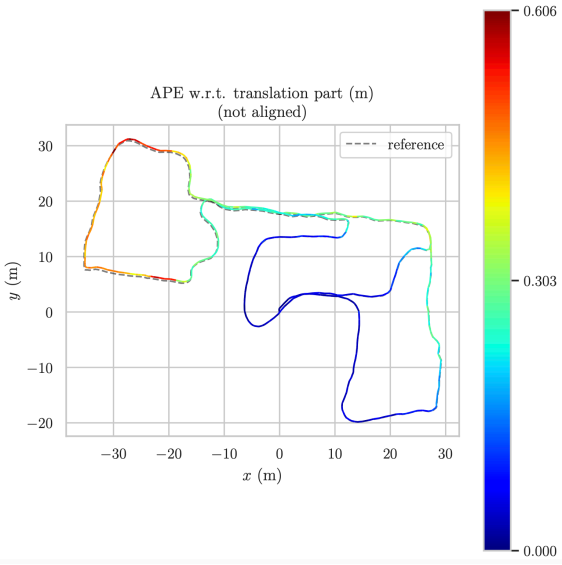


(b) Trajectories viewed in a function of time for the *yaw*, *pitch* and *roll* rotations

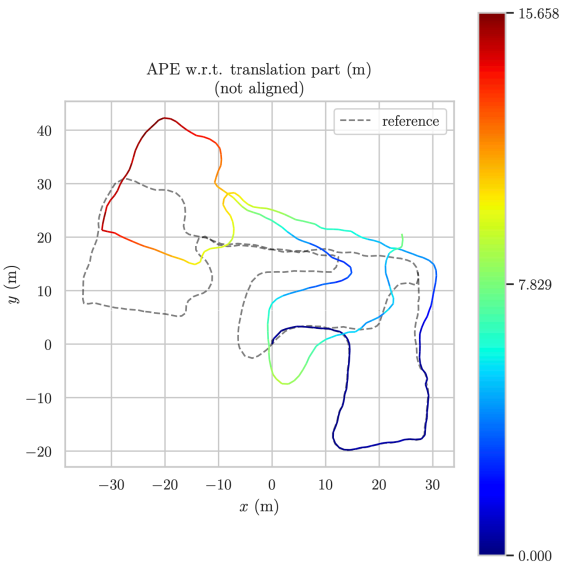


(c) The xy-plane view of the trajectories

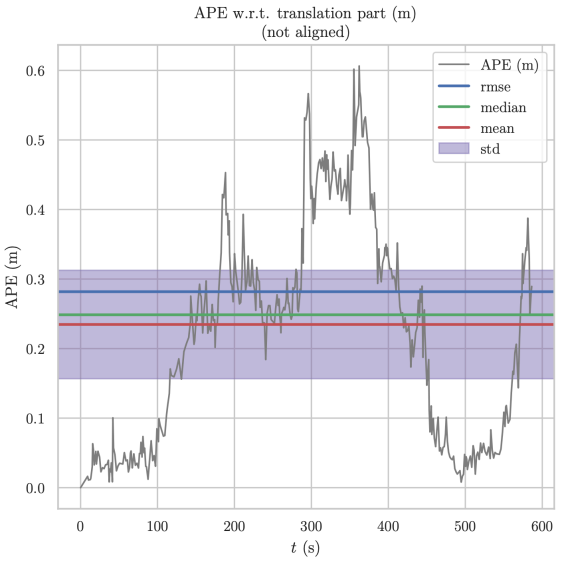
Fig. 7. Visualization of the sequence five's the trajectories obtained from A-LOAM and LIO-SAM against our ground-truth trajectory



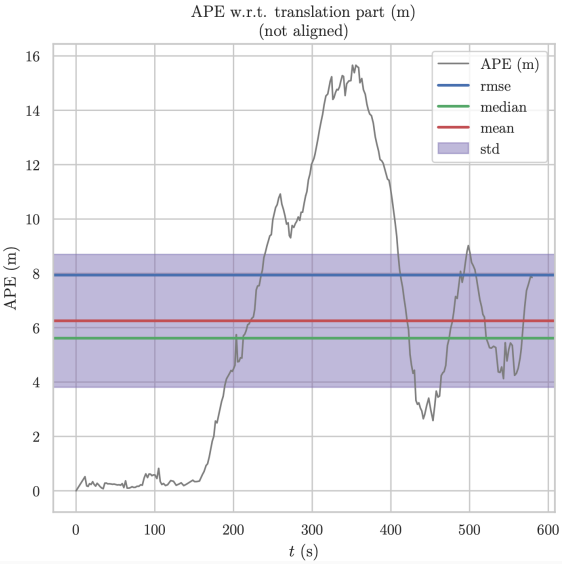
(a) Color-coded APE for LIO-SAM viewed in the xy-plane



(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