ConSLAM: Periodically Collected Real-World Construction Dataset for SLAM and Progress Monitoring

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

ConSLAM: Periodically Collected Real-World Construction Dataset for SLAM and Progress Monitoring

Maciej Trzeciak[1] (✉), Kacper Pluta[2], Yasmin Fathy[1] Lucio Alcalde[3], Stanley Chee[3], Antony Bromley[3], Ioannis Brilakis[1], and Pierre Alliez[2]

[1] Department of Engineering, University of Cambridge, UK
`{mpt35,yafa2,ib340}@cam.ac.uk`
[2] Inria Sophia Antipolis-Méditerranée, France
`{kacper.pluta,pierre.alliez}@inria.fr`
[3] Laing O'Rourke, Dartford DA2 6SN, UK
`{lalcalde,schee,abromley}@laingorourke.com`

Abstract. Hand-held scanners are progressively adopted to workflows on construction sites. Yet, they suffer from accuracy problems, preventing them from deployment for demanding use cases. In this paper, we present a real-world dataset collected periodically on a construction site to measure the accuracy of SLAM algorithms that mobile scanners utilize. The dataset contains time-synchronised and spatially registered images and LiDAR scans, inertial data and professional ground-truth scans. To the best of our knowledge, this is the first publicly available dataset which reflects the periodic need of scanning construction sites with the aim of accurate progress monitoring using a hand-held scanner.

Keywords: real-world dataset, SLAM, construction progress monitoring, point cloud

1 Introduction

The digitization of the geometry of existing infrastructure assets is a crucial step for creating an effective *Digital Twin* (DT) for many applications in *Architecture, Engineering and Construction* (AEC) industry. On the one hand, the growing adoption of mobile and hand-held scanning devices brings the hope of increased productivity with respect to capturing the geometric data. On the other, however, the underlying *Simultaneous Localization And Mapping (SLAM)* algorithms, which such scanners utilize, are not yet accurate enough to meet the requirements of demanding use cases such as engineering surveying. As a result, there is a mix of technologies used on construction sites.

Publicly available datasets serve as battlegrounds against which different methods compare their performance. Yet, there are very few of them that would enable the comparison of SLAM methods on construction sites. In fact, there is no publicly available dataset that would reflect the periodic need of scanning construction sites, with the view of accurate progress monitoring using a hand-held scanner. Our paper aims at addressing this problem.

Under the following link `https://github.com/mac137/ConSLAM`, we present a real-world dataset, the "**ConSLAM**", recorded by our prototypical hand-held scanner.

The dataset consists of four sequences captured at the same floor of a construction site. We recorded one sequence approximately every month. Each sequence contains *Red-Green-Blue* (RGB) and *Near-InfraRed* (NIR) images of resolutions 2064×1544 and 2592×1944 pixels respectively, 16-beam Velodyne LiDAR scans and 9-axis *Inertial Measurement Unit* (IMU) data. The first three sensors were synchronised in time and recorded at about 10 Hz while IMU was recorded at about 400 Hz. The acquired sequences vary in their duration between five and nine minutes. For every sequence, we also include a *Ground-Truth* (GT) point cloud provided by a land surveying team. We used these point clouds to produce the ground-truth trajectories of our scanner, against which SLAM algorithms can measure their accuracy. Our hope is that all these modalities will enable further exploration of mobile mapping algorithms.

This paper is structured as follows. Section 2 provides a discussion on the existing datasets. Section 3 includes the description of our hand-held prototype, as well as other devices and methods we used to produce the complete dataset. In Section 4, we briefly describe the construction site and present the structure as well as availability of our dataset. We close the paper by discussing future steps in Section 5.

2 Existing datasets

Mobile scanning systems are portable devices that integrate multiple sensors for obtaining detailed surveys of scanned scenes by creating 3D point cloud data. There are different sequential point cloud datasets, but very few, such as the Hilti SLAM challenge dataset [12] are both sequential as well as collected for construction sites.

The available sequential datasets can be categorized into two main types: synthetic and real-world. A synthetic dataset is artificially generated in a virtual world by simulating a real-world data acquisition system. A sequential dataset is collected as sequences of frames from a movable platform, e.g., vehicular or hand-held ones. Most studied sequential datasets are described in this section, and also summarized in Table 1.

KITTI[4] is a well-known benchmark collected mainly for 3D object detection scenarios [8, 9]. The data includes six hours of traffic scenarios at 10–100 Hz using a system mounted on a moving vehicle with a driving speed up to 90 km/h. The system comprises data from high-resolution colour and greyscale stereo cameras, a LiDAR, a *Global Positioning System* (GPS) as well as IMU devices [8]. The set-up allows collecting data that are suitable for different tasks: stereography, optical flow, *Visual Odometry* (VO) and 3D object detection. For the visual odometry benchmark, the data contain 22 sequences of images, 11 of them being associated to ground-truth, and the remaining mainly contains raw sensor data. The ground-truth for VO is the output of GPS/IMU localization. The data is also provided along with the trajectories.

SemanticKITTI [1] is based on the KITTI dataset, mainly the sequences provided for the OV task. SemanticKITTI provides dense point-wise annotation for zero to ten sequences, while the other 11-21 sequences are used for testing, making the data suitable for various tasks. Three main tasks are proposed for SemanticKITTI; semantic segmentation of a scene, semantic scene completion (i.e., predicting future semantic scenes), and semantic segmentation of multiple sequential scans.

[4] http://www.cvlibs.net/datasets/kitti

SemanticPOSS [14] is a dataset that contains LiDAR scans with dynamic instances. It uses the same data format as SemanticKITTI. Similar to KITTI, SemanticPOSS has been collected by a moving vehicle equipped with a Pandora module[5] and a GPS/IMU localization system to collect 3D point cloud data. The Pandora integrates cameras and LiDAR into the same module. The vehicle travelled a distance of around 1.5 kilometres on a road that includes many moving vehicles and walking and riding students. The collected data are annotated that each point contains unique instance labels for dynamic objects (car, people, rider). The data are suitable for predicting the accuracy of dynamic objects and people [7] and 3D semantic segmentation [14]. The data size of Semantic-POSS is limited compared to SemanticKITTI. Although there is a higher resolution on horizontal LiDAR scans, the spatial distribution of the LiDAR points is unbalanced [7].

SynthCity [10] is a synthetic labelled point cloud dataset generated from a synthetic full-colour mobile laser scanning with a predefined trajectory. Each point is labelled by one of nine categories: high vegetation, low vegetation, buildings, scanning artefacts, cars, hardscape, man-made terrain, and natural terrain. The synthetic point clouds are generated in urban/suburban environments modelled within Blender 3D graphics software[6]. The dataset has been released primarily for semantic per-point classification, where each point contains a local feature vector and a classification label. However, the dataset is unsuitable for instance segmentation as it does not include instances' identifiers.

The Grand Theft Auto V (GTA5) [15] is a synthetic sequential point cloud dataset that was generated based on the photo-realistic virtual world in the commercial video game "Grand Theft Auto V". The approach is based on creating large-scale pixel-level semantic segmentation by extracting a set of images from the game and then applying a pipeline to produce the corresponding label. The game includes different resource types, including texture maps and geometric meshes, combined to compose a scene, which facilitates establishing the associations between scene elements. GTA5 is three orders of magnitude larger than semantic annotations included in the KITTI dataset [8, 15]. The data contains 19 semantic classes, including road, building, sky, truck, person, traffic light and other objects on road scenes. The data was used for training semantic segmentation models and evaluated on two datasets, including KITTI [8] where the training phase included both real and synthetic data using minibatch stochastic gradient descent. The model trained with generated synthetic data within GTA5 outperforms the model trained without it by factor 2.6.

The nuTonomy scenes (nuScenes) [4] is a real-world dataset for collecting point cloud using six cameras, five radars and a LiDAR, each with a full 360-degree field of view. The data is fully annotated with 3D bounding boxes, mainly for autonomous driving scenarios, with available map information associated to the collected data. The data include trajectories as idealized paths that the movable platform should take; assuming there are no obstacles in the route. The data include 23 classes: road, pavement, ground, tree, building, pole-like, and others. Compared to the KITTI dataset, nuScenes has seven times more object annotations and one hundred times more images. The dataset is

[5]Please consult www.hesaitech.com

[6]https://www.blender.org/

currently suitable for 3D object detection and tracking, where tracking annotation is also available [5].

The Hilti SLAM 2021 challenge dataset [12] is a benchmark dataset that collects multiple sensor modalities of mixed indoor and outdoor environments with varying illumination conditions and along the trajectory. The indoor sequences portray labs, offices and construction environments, and the outdoor sequences were recorded in parking areas and on construction sites[7]. The data were collected by a handheld platform comprising multiple sensors: five AlphaSense cameras (stereo pair), two LiDARs (Ouster OS0-64 and Livox MID70), and three IMUs (ADIS16445) with accurate spatial and temporal calibration. The main aim of this dataset is to promote the development of new SLAM algorithms that attain both high accuracy and robustness for challenging real-world environments such as construction sites. In 2022, the same challenge was organized, but the data were collected mainly for construction sites and Sheldonian Theatre in Oxford, UK[8]. The data were collected by a sensor suite mounted on an aluminium platform for handheld operation. The suite consists of a Hesai PandarXT-32 and Sevensense Alphasense Core camera head with five 0.4MP global shutter cameras. The LiDAR and cameras are synchronised via *Precision Time Protocol* and all sensors are aligned within one millisecond.

To the best of our knowledge, the **ConSLAM** dataset is the first sequential dataset with a trajectory for a construction site that aims to capture the construction of a site over a few months. The data are collected by a hand-held mobile scanner comprising a LiDAR, RGB and NIR cameras and an IMU (shown in Figure 1). All the modalities are synchronised in time and spatially registered. This aims to foster novel research and progress in evaluating SLAM approaches and trajectory tracking at construction sites and developing progress monitoring and quality control systems for the AEC community.

3 Methodology

In this section, we introduce the configuration of the hand-held prototype device, and our data acquisition and post-processing pipelines.

3.1 Sensors and devices

The sensors used during data acquisition at the construction site are shown in Figure 1. As shown in Figure 1(a), our prototypical hand-held scanner consists of a LiDAR at the top (Velodyne VLP-16), an RGB camera (Alvium U-319c, 3.2 MP) located directly below the LiDAR, a NIR camera (Alvium 1800 U-501, 5.0 MP) located to the right of the RGB camera and an IMU (Xsens MTi-610), to the left of the RGB camera. They are rigidly attached to a custom-made aluminium frame with a handle at the bottom. All of them are connected to a laptop (MacBook Pro 2021) where data were recorded and pre-processed. In addition, a Leica RTC 360[9] (see Figure 1(b)) is used to collect

[7]https://hilti-challenge.com/dataset-2021.html

[8]https://hilti-challenge.com/dataset-2022.html

[9]https://leica-geosystems.com/products/laser-scanners/scanners/leica-rtc360

Table 1. Summary of existing sequential point cloud datasets

Name	**Real/ Synthetic**[a]	**Indoor/ Outdoor** [b]	**Trajectory** [c]	**Sector**	**Applications**	**Sensors**
KITTI [9]	R	I	✓	Urban and transport	Autonomous vehicles [6, 13], 3D object detection and visual odometry [9]	Four colour and greyscale stereo cameras, a laser scanner (Velodyne), four Edmund optics lenses, GPS navigation systems
Semantic-KITTI [1]	R	O	✓	Urban/road	Semantic segmentation of a scene, semantic scene completion (i.e., predicting future semantic scenes), and semantic segmentation of multiple sequential scans [1]	Relying on the data collected by laser scanner (Velodyne) in the KITTI dataset
HILTI-OXFORD [12]	R	I &O	✓	Construction	Construction robotics, construction site environments	Five AlphaSense cameras (stereo pair), two LiDARs (Ouster OS0-64 and Livox MID70), and three IMUs (ADIS16445)
Semantic-POSS [14]	R	O	✗	Urban/road	Prediction accuracy of dynamic objects and people [7] and 3D semantic segmentation [14]	Pandora module (LiDAR, mono and colour cameras) and GPS/IMU
SynthCity [10]	S	O	✓	Urban/ suburban environments	Point cloud classification [10]	Mobile laser scanning
GTA5 [15]	S	O	✗	Urban/road	Semantic segmentation and scene understanding [15]	Frames extracted from "Grand Theft Auto V" video game; from a car perspective
nuScenes [4]	R	O	✓	Urban/road and autonomous driving	Object detection and tracking, segmentation [4]	Six cameras, five radars and one LiDAR, all with full 360 degree field of view
ConSLAM	R	O	✓	Construction	Progress monitoring & quality control, object detection and tracking	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)

[a] R: real-world, S: synthetic/artificial
[b] I: indoor, O: outdoor
[c] Indicates whether the data includes sensor's path, i.e., trajectory.

precise scans which are later stitched together and geo-referenced by land surveyors. These scans serve as our ground-truth.

3.2 Intrinsic calibrations of the sensors

Both cameras, i.e., RGB and NIR, are intrinsically calibrated according to the Brown-Conardy model [3] with the camera intrinsic matrix and lens distortion coefficients stored along with the dataset. The resolution of distorted images is 2064×1544 and 2592×1944 pixels for the RGB and NIR cameras, respectively. Moreover, the LiDAR's intrinsic parameters have mostly default values as in the manufacturer's man-

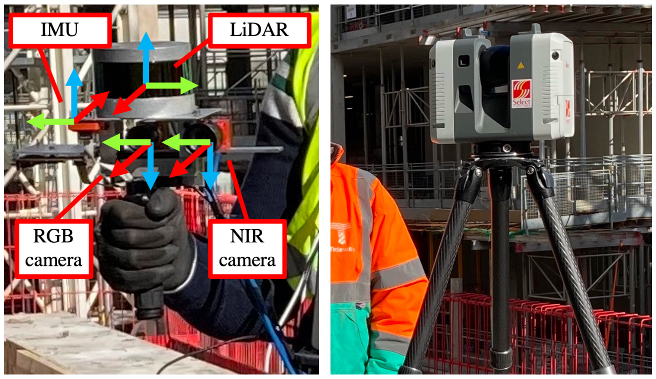


Fig. 1. Data acquisition: (a) our prototypical hand-held scanner, and (b) a static scanner used to collect ground-truth scans. Red, green and blue colours represent X-, Y- and Z-axes, respectively

ual and the Velodyne driver[10] with a restricted range of 60 meters and the parameter `lidar_timestamp_first_packet` is set to `true`.

3.3 Extrinsic calibrations of the sensors

The LiDAR sensor is extrinsically calibrated in a pair with other sensors as follows: the LiDAR and the RGB camera, the LiDAR and the NIR camera, and the LiDAR and the IMU. The sensor frames are positioned against each other in our prototypical hand-held scanner, as shown in Figure 1(a). It is worth-mentioning that when the scanner is held vertically (i.e., in its operational position), LiDAR's X-, RGB camera's Z- and NIR camera Z-axes face the front while IMU's X-axis faces backward. LiDAR's and IMU's Z-axes face upwards, while the RGB and NIR cameras' Y-axes face downwards. The remaining axes can be further deduced from the figure.

We used a method proposed by Beltrán et al. [2] to extrinsically calibrate the LiDAR with both the RGB camera and the NIR camera. We used the method used by VINS-Mono[11] for LiDAR-IMU calibration. Their respective matrices are stored along with the dataset.

3.4 Data collection system of the hand-held scanner

Our hand-held data collection system utilizes *Robot Operating System* (ROS)[12] as a backbone to process the data streams coming from all four sensors. Figure 2 presents the data processing pipeline in more detail.

[10] `https://github.com/ros-drivers/velodyne`

[11] see `https://github.com/chennuo0125-HIT/lidar_imu_calib`, and `https://blog.csdn.net/weixin_37835423/article/details/110672571`

[12] `https://www.ros.org`

We first launch the respective drivers of the four sensors, thus publishing the individual data messages to our ROS-based system as shown in the top layer in Figure 2. Next, we synchronize the RGB camera, LiDAR and NIR camera in time using a standard ROS synchronization policy[13], based on matching messages whose difference in timestamps is smaller than 10 milliseconds. However, we decided to split this process into two because of problems encountered with our NIR camera. If the synchronization was matching timestamps from all three topics and any of the topics stopped working for a moment, the synchronization of all the three topics would stop too. Our NIR camera sporadically stops publishing images for a moment, which would effectively stop the synchronization of all three sensors. Instead, we decided to synchronise RGB images and LiDAR scans first and publish them on `/pp_rgb/synced2points` and `/pp_points/synced2rgb` topics respectively. NIR images are then synchronised with `/pp_points/synced2rgb` using the same synchronization policy and published on the `/pp_nir/synced2points` topic. This solution allows us to keep recording synchronized RGB images and lidar scans even when the NIR camera stops working for a moment.

During scanning, we also monitor the three synchronised topics and the IMU messages to make sure that our system actually receives data from the sensors. In the last step, we record the synchronised topics along with the IMU data (`/imu/data`) and store

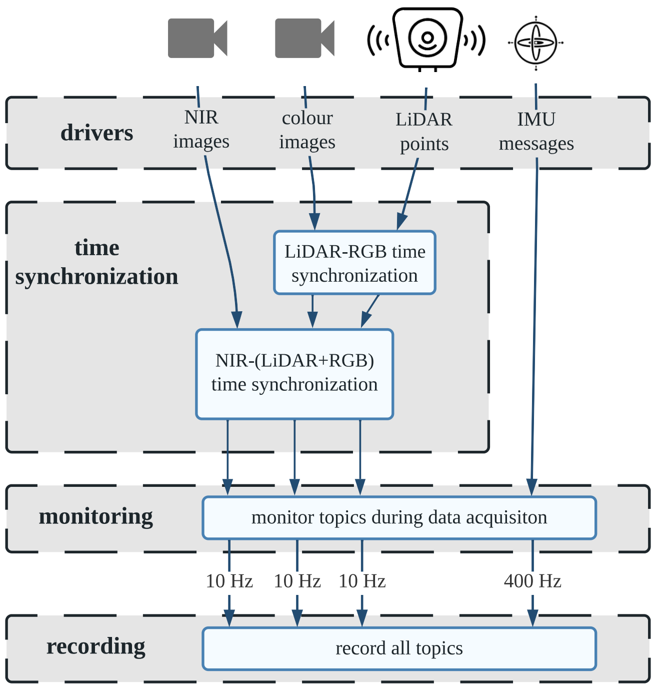


Fig. 2. Processing data streams on construction site

[13]https://wiki.ros.org/message_filters/ApproximateTime

them as a standard bag file. We decided to record IMU messages at 400 Hz because it is often the case that higher IMU rate improves the performance of SLAM algorithms [16].

3.5 Ground-truth trajectories

This section discusses the ground-truth dataset and focuses on our post-processing pipeline used for its creation and refinement. We also discuss the issue of registering individual LiDAR scans to the ground-truth dataset to recover the ground-truth trajectory of our prototypical scanner. The ground-truth dataset contains four scans referred to as $GT_i, i = 1, \dots, 4$, which were collected over a period of three months on an active construction site, as explained in Section 4.

Registration Error Metric: we faced two main issues during the registration process of datasets: varying overlap and geometric discrepancies. Such issues make a 3D point cloud registration difficult and require manual adjustments in order to ensure high precision of the data alignment. Such a registration can take up to a few dozens of minutes to be properly performed by an experienced person.

Nevertheless, we need to be able to provide a registration error metric for the datasets, hampered by the aforementioned issues. There exist several error metrics, which can be used for our purposes. We opted 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 $S_d = \{q \mid \|\gamma(q) - q\| < d\} \subset P_{target}$.

In our experiments, the threshold distance d is empirically set to one centimetre.

Registration of static scans: each of the GT_i sets, was obtained from a multi-view registration of M_i scans. In this section, we define the multi-view registration problem.

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 $\mathbb{G} = \{g_k \mid 1 \leq k \leq M, g_k \in \mathrm{SE}(3)\}$ be a set of rigid transformations. The multi-view registration problem can be then formulated as

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

where $\{p_k^j \to q_k^j\}$ 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

Table 2. RMSE$_d$ distance for ground-truths dataset. The measurements are recorded in centimetres

Dataset name	$\approx \min_{\mathrm{RMSE}_d}$	$\approx \max_{\mathrm{RMSE}_d}$	$\approx \mathrm{mean}_{\mathrm{RMSE}_d}$
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

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 [11].

Having registered the scans obtained from a terrestrial scanner, we have downsampled them using distance-based downsampling[14] with the threshold of five millimetres. Finally, we compute RMSE$_d$ distances between overlapping point sets, see Table 2.

Registration of LiDAR scans to ground-truth scans: we create a ground-truth trajectory of the LiDAR by registering each LiDAR scan to the ground-truth scans. This means that we recover the true pose of each LiDAR scan with respect to the ground-truth scans received from land surveyors. To do that, we *play* each recorded bag file and save every LiDAR message to the PLY file format as an individual LiDAR scan. In addition, we run *Advanced Lidar Odometry and Mapping* (A-LOAM[15]) algorithm—an implementation of the LOAM algorithm proposed by Zhang et al. [17]—on each bag file and save odometric poses corresponding to the individual LiDAR scans as text files. The text files and the individual LiDAR scans are then matched based on their timestamps assigned during the data collection.

The ICP algorithm is then executed on each pose-scan pair. We extract edges from every LiDAR scan in the same way as it is done in A-LOAM and use the corresponding pose as an initial guess for the fine registration. The ICP is performed for a couple of iterations, starting with half a metre as an initial threshold for establishing correspondences to the closest points. With every iteration, the threshold decays by a factor of 0.85 and the algorithm converges when the RMSE/fitness is smaller than three centimetres.

The resulting six Degree of Freedom (6-DOF) transformations are stored as 4×4 matrices and are named after the corresponding LiDAR scans. In other words, these matrices represent the transformations that the LiDAR scans must undergo to be aligned with the ground-truth scans. The whole collection of transformations makes up the ground-truth trajectory. Figures 4 and 5 present photorealistic renderings of the ground-truth data and the position of the registered LiDAR scans.

We implemented a computer program to automate the registration process described above. However, there were still situations where our software was unsuccessful. This includes the following: (1) the drift by the LOAM algorithm run on the stream of LiDAR scans was high enough that its poses fed to our registration algorithm were too distant

[14]We used the distance-based downsampling implemented in CloudCompare 2.12.2. See https://www.cloudcompare.org.

[15]https://github.com/HKUST-Aerial-Robotics/A-LOAM

to find correct correspondences between the extracted LiDAR features and the ground truth scans; (2) the exact trajectory our scanner followed is not fully covered by the ground-truth scans from the land surveying team, hence it was not possible to align the LiDAR scans from such places to the ground-truth scans. An example relating to the first issue can be seen in Figure 3. We estimate that we have correctly registered around 80% of LiDAR scans to GT_1, about 80% of LiDAR scans to GT_2, approximately 60% of LiDAR scans to GT_3 and roughly 70% of LiDAR scans to GT_4.

4 Dataset: ConSLAM

We collected the four raw streams of data at a part of a story at Whiteley's in London which had been originally designed by John Belcher and John James Joass in 1911 as one of London's leading department stores. At the time of writing this paper, the building is undergoing redevelopment, which involved the demolition of the existing shopping centre behind a retained historic façade. The new development involved the creation of luxury retail, leisure and a residential scheme involving the construction of a new six to nine-story building.

4.1 Dataset structure

Our dataset is structured as shown in Figure 6. The directory includes five main files in the ZIP file format, four of them containing data from four individual scans carried once per month. The scans are numbered from 1 to 4, with the earliest scan marked with 1 and the oldest one marked with 4. In order to save space in Figure 6, we encoded this fact with `data_unpacked_x.zip` where $x = 1, \dots, 4$.

Each of the four data zipped files contains a `recording.bag` file recorded during scanning. This file can be *played* using rosbag[16] and contains four topics with the stream of RGB and NIR images, LiDAR points and IMU messages. `groundtruth_scan.ply` file is our ground-truth point cloud created by land surveyors as described in Section 3.5. Next, there are three folders `rgb/`, `nir/` and `lidar/` which contain messages unpacked

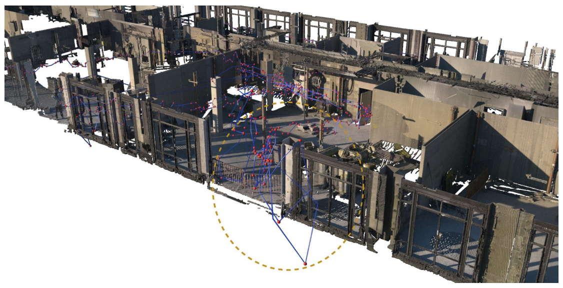
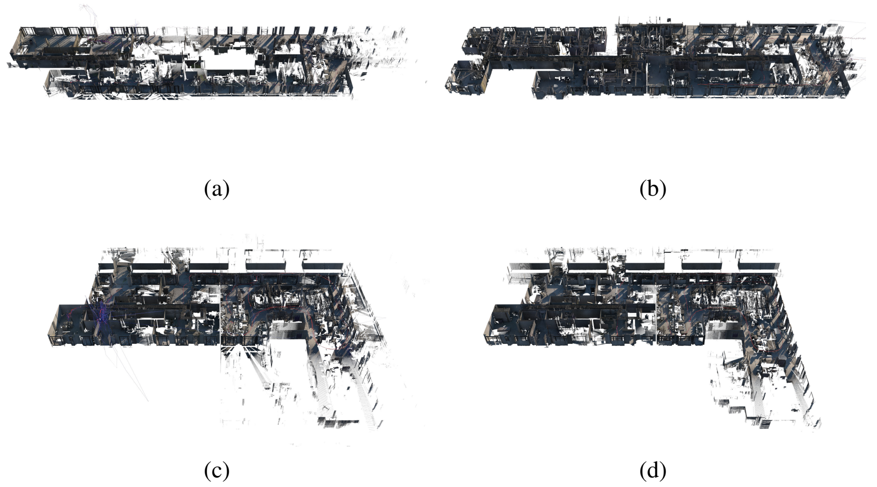


Fig. 3. Visualization of incorrectly registered LiDAR poses, which have been marked by the light-brown ellipse

[16]https://wiki.ros.org/rosbag



(a) (b) (c) (d)

Fig. 4. Top-view visualization of the ground-truth datasets: GT_1 – (a), GT_2 – (b), GT_3 – (c), GT_4 – (d). Note that in some places we can see incorrectly registered LiDAR poses, i.e., (a) and (c)

from the `recording.bag` file. The corresponding LiDAR scans in `lidar/` and images in `rgb/`, `nir/` are named with the same timestamp coming from LiDAR scans recorded during scanning. The last folder `pose/` contains the ground-truth poses of the LiDAR sensor created as described in Section 3.5 and named also with the corresponding timestamps. The collection of these poses makes up the ground-truth trajectory of the LiDAR sensor.

The file `data_calib.zip` includes all the calibration parameters including: RGB and NIR camera calibration matrices along with their distortion coefficients in `calib_rgb.txt` and `calib_nir.txt` respectively. Moreover, there is a rigid-body transformation matrix between the LiDAR and the RGB camera in `calib_lidar2rgb.txt`, rigid-body transformation matrix between the LiDAR and the NIR camera in `calib_lidar2nir.txt`, and finally, a rotation matrix between the LiDAR and IMU in `calib_lidar2imu.txt`.

4.2 Practical application: Projecting LiDAR points onto corresponding images

Our **ConSLAM** dataset is available at https://github.com/mac137/ConSLAM, where a complete description and examples on how to use the data are provided.

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

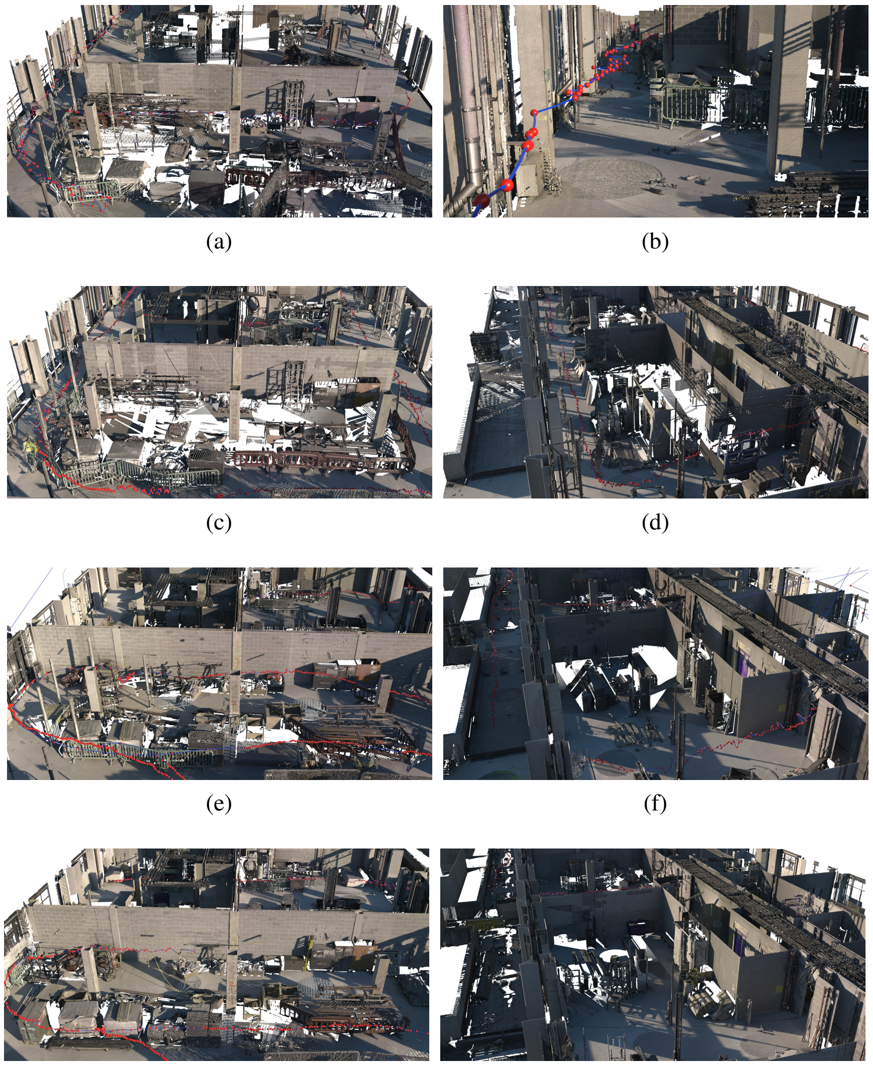
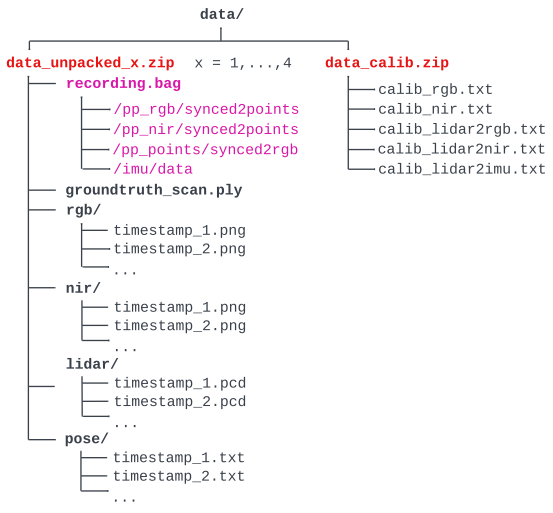

(a) (b) (c) (d) (e) (f) (g) (h)

Fig. 5. Close-up visualization of the ground-truth datasets: GT_1 – (a-b), GT_2 – (c-d), GT_3 – (e-f), GT_4 – (g-h), together with the LiDAR positions depicted by red spheres. The LiDAR position have been connected to provide approximated paths



```
                               data/
        ┌────────────────────────┴────────────────────────┐
data_unpacked_x.zip  x = 1,...,4                   data_calib.zip
├── recording.bag                                  ├──calib_rgb.txt
│       ├──/pp_rgb/synced2points                   ├──calib_nir.txt
│       ├──/pp_nir/synced2points                   ├──calib_lidar2rgb.txt
│       ├──/pp_points/synced2rgb                   ├──calib_lidar2nir.txt
│       └──/imu/data                               └──calib_lidar2imu.txt
├── groundtruth_scan.ply
├── rgb/
│       ├──timestamp_1.png
│       ├──timestamp_2.png
│       └──...
├── nir/
│       ├──timestamp_1.png
│       ├──timestamp_2.png
│       └──...
├── lidar/
│       ├──timestamp_1.pcd
│       ├──timestamp_2.pcd
│       └──...
└── pose/
        ├──timestamp_1.txt
        ├──timestamp_2.txt
        └──...
```

Fig. 6. Dataset folder structure

$\mathbf{K}^{\text{RGB}}_{undist}$ using OpenCV[17] package, $\mathbf{K}^{\text{RGB}}_{dist}$ and $(k_1, k_2, k_3, k_4, k_5)$. We find the corresponding LiDAR scan in the `lidar/` folder using the file name of the image, and we iterate over its points. In order to project a single LiDAR point $\mathbf{x}_i = [x_i, y_i, z_i]^\top$ onto the undistorted images, we follow

$$\begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = \mathbf{K}^{\text{RGB}}_{undist} \begin{bmatrix} \mathbb{1}_{3\times3} \\ \mathbf{0}_{1\times3} \end{bmatrix}^\top \mathbf{T}^{\text{LiDAR}^{-1}}_{\text{RGB}} \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix}, \tag{4}$$

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 7, which shows an example of LiDAR points projected onto the corresponding image.

5 Conclusion and future direction

We introduced a new real-world dataset, the "**ConSLAM**", recorded periodically by a hand-held scanner on a construction site. The dataset aims at facilitating the comparison of SLAM algorithms for periodic and accurate progress monitoring in construction. The dataset contains the ground-truth trajectories of the scanner, which allows for an accurate comparison of SLAM methods run on our recorded data streams.

[17]https://opencv.org

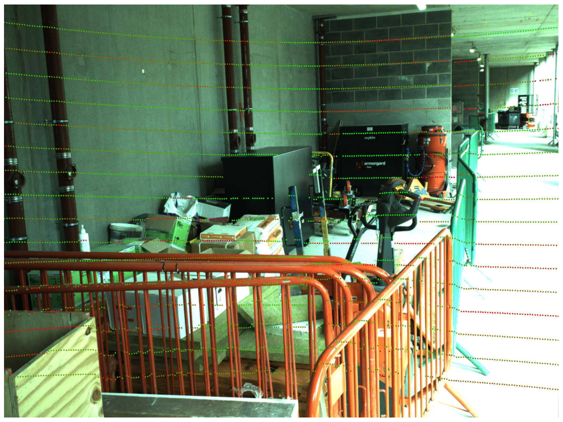


Fig. 7. Example of projecting of LiDAR points onto the corresponding image

In the future, we aim at registering of all LiDAR scans to the ground-truth scans, hence recovering the full ground-truth trajectory of our scanner. The objective is also to extend this dataset with ground-truth for semantic segmentation of images and point clouds. This will allow the development of progress monitoring systems based on the comparison of Design-Intent and As-Built, for example, using the volume of individual building elements. The semantic annotations will also allow prospective algorithms to measure the accuracy of inferred information in such popular tasks like object detection or instance segmentation.

Acknowledgment: the authors would like to thank Laing O'Rourke for allowing access to their construction site and for collecting the ground truth scans. We also acknowledge Romain Carriquiry-Borchiari of Ubisoft France for his help with rendering some of the figures. This work is supported by the EU Horizon 2020 BIM2TWIN: Optimal Construction Management & Production Control project under an 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

1. Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Stachniss, C., Gall, J.: Semantickitti: A dataset for semantic scene understanding of lidar sequences. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 9297–9307 (2019)
2. Beltrán, J., Guindel, C., de la Escalera, A., García, F.: Automatic extrinsic calibration method for lidar and camera sensor setups. IEEE Transactions on Intelligent Transportation Systems (2022). https://doi.org/10.1109/TITS.2022.3155228
3. Brown, D.: Decentering distortion of lenses. In: Photogrammetric Engineering. pp. 444–462 (1966)
4. Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuscenes: A multimodal dataset for autonomous driving. In: Proceedings

of the IEEE/CVF conference on computer vision and pattern recognition. pp. 11621–11631 (2020)
5. Chang, M.F., Lambert, J., Sangkloy, P., Singh, J., Bak, S., Hartnett, A., Wang, D., Carr, P., Lucey, S., Ramanan, D., et al.: Argoverse: 3d tracking and forecasting with rich maps. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8748–8757 (2019)
6. Fritsch, J., Kuehnl, T., Geiger, A.: A new performance measure and evaluation benchmark for road detection algorithms. In: 16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013). pp. 1693–1700. IEEE (2013)
7. Gao, B., Pan, Y., Li, C., Geng, S., Zhao, H.: Are we hungry for 3d lidar data for semantic segmentation? a survey and experimental study. arXiv preprint arXiv:2006.04307 (2020)
8. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: The kitti dataset. The International Journal of Robotics Research **32**(11), 1231–1237 (2013)
9. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: 2012 IEEE conference on computer vision and pattern recognition. pp. 3354–3361. IEEE (2012)
10. Griffiths, D., Boehm, J.: Synthcity: A large scale synthetic point cloud. arXiv preprint arXiv:1907.04758 (2019)
11. Haarbach, A.: Multiview ICP, http://www.adrian-haarbach.de/mv-lm-icp/docs/mv-lm-icp.pdf
12. Helmberger, M., Morin, K., Kumar, N., Wang, D., Yue, Y., Cioffi, G., Scaramuzza, D.: The hilti slam challenge dataset. arXiv preprint arXiv:2109.11316 (2021)
13. Menze, M., Geiger, A.: Object scene flow for autonomous vehicles. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3061–3070 (2015)
14. Pan, Y., Gao, B., Mei, J., Geng, S., Li, C., Zhao, H.: Semanticposs: A point cloud dataset with large quantity of dynamic instances. In: 2020 IEEE Intelligent Vehicles Symposium (IV). pp. 687–693. IEEE (2020)
15. Richter, S.R., Vineet, V., Roth, S., Koltun, V.: Playing for data: Ground truth from computer games. In: European conference on computer vision. pp. 102–118. Springer (2016)
16. Shan, T., Englot, B., Meyers, D., Wang, W., Ratti, C., Daniela, R.: Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 5135–5142. IEEE (2020)
17. Zhang, J., Singh, S.: Loam: Lidar odometry and mapping in real-time. In: Proceedings of Robotics: Science and Systems. Berkeley, USA (July 2014). https://doi.org/10.15607/RSS.2014.X.007