INITIAL INVESTIGATION OF A LOW-COST AUTOMOTIVE LIDAR SYSTEM

A. Ortiz Arteaga, D. Scott, J. Boehm*

Dept. of Civil, Environmental & Geomatic Engineering, University College London, Gower Street, London, WC1E 6BT UK
angelica.arteaga.17@ucl.ac.uk, d.scott@ucl.ac.uk, j.boehm@ucl.ac.uk

Commission II

KEY WORDS: LIDAR, Low-Cost, Automotive

ABSTRACT:

This investigation focuses on the performance assessment of a low-cost automotive LIDAR, the Livox Mid-40 series. The work aims to examine the qualities of the sensor in terms of ranging, repeatability and accuracy. Towards these aims a series of experiments were carried out based on previous research of low-cost sensor accuracy, LIDAR accuracy investigation and TLS calibration experiments. The Livox Mid-40 series offers the advantage of a long-range detection beyond 200 m at a remarkably low cost. The preliminary results of the tests for this sensor indicate that it can be used for reality capture purposes such as to obtain coarse as-built plans and volume calculations to mention a few. Close-range experiments were conducted in an indoor laboratory setting. Long-range experiments were performed outdoors towards a building façade. Reference values in both setups were provided with a Leica RTC 360 terrestrial LIDAR system. In the close-range experiments a cross section of the point cloud shows a significant level of noise in the acquired data. At a stand-off distance of 5 m the length measurement tests reveal deviations of up to 11 mm to the reference values. Range measurement was tested up to 130 meters and shows ranging deviations of up to 25 millimetres. The authors recommend further investigation of the issues in radiometric behaviour and material reflectivity. Also, more knowledge about the internal components is needed to understand the causes of the concentric ripple effect observed at close ranges. Another aspect that should be considered is the use of targets and their design as the non-standard scan pattern prevents automated detection with standard commercial software.

1. INTRODUCTION

1.1 General Introduction

Laser scanning has become a very commonly used method for reality capture, gaining popularity in many fields such as the architecture, engineering and construction (AEC) sector. A primary reason for this technique's popularity is due to its many applications. Some of these applications include deformation monitoring, High Definition Surveying (HDS), 3-Dimensional (3D) modelling and capturing data for the Building Information Modelling (BIM) process (Thomson and Boehm, 2014). Moreover, Reality Capture (RC), has become a vital asset for collecting data in new industries. Examples of these comprehend autonomous vehicles, virtual and augmented reality amongst others. The incorporation of this technology is due to an ever-growing demand for agents to be aware and understand their surroundings. Even for industries that do not collect data on first-hand but rely on extensive public/private datasets, i.e. environmental monitoring.

In recent years considerable efforts have been spent for creating the autonomous car of the future. Many interested parties see state-of-the-art LIDAR technology as a key technology for the automotive industry. Several established companies and a multitude of start-up companies have developed and adopted LIDAR technology into vehicles for navigation. Autonomous vehicles could navigate in constrained environments surrounded by mobile objects hence, they have to continuously observe the drivable space to enrich passengers and third person safety (Moras et al., 2012).

Although this new generation of low-cost, small LIDAR sensors may not have the same standards as established surveying equipment, they should not be overlooked, as they could still provide interesting solutions for the AEC industries.

In the United Kingdom, the Royal Institution of Chartered Surveyors (RICS) recommendations for professional surveying tasks state that not all projects require the same level of accuracy (Royal Institution of Chartered Surveyors, 2014). This means that different methods and devices can be utilised for a job. Therefore, not all tasks require the same amount of resources. Typically, the cost of a LIDAR or Terrestrial Laser Scanner (TLS) is influenced by the accuracy that it can deliver. The rise of novel low-cost scanner extends the options of tools that can be used for RC, unveiling a niche in the market for the RC and AEC industries and creating the possibility of increasing direct costumers' profitability and deliver a custom-tailored service to end-users of LIDAR-derived products.

Besides the cost of an instrument and its accuracy requirements, the resolution of a LIDAR is another matter of importance, as this aspect affects the level of details that can be observed from a point cloud. As a consequence, this condition has a direct implication on the quality of the 3D model that could be obtained (Ling et al., 2008). Additionally, previous research of low-cost 3D sensors mentions that accuracy and repeatability are crucial when assessing structured- light-based 3D sensors (Boehm, 2014).

1.2 Introduction to the Livox Mid-40 series

In this study we want to investigate the capabilities of a new automotive LIDAR system that became available to the market in 2019 for only $600. This is significantly below the cost of establishes automotive LIDAR systems which often have a price that is an order of magnitude higher. The Livox Mid-40 series

* Corresponding author

Figure 1. Livox Mid-40 LIDAR sensor

Laser Wavelength	905 nm
Detection Range	90 m @ 10% reflectivity 130 m @ 20% reflectivity 260 m @ 80% reflectivity
Field of view	38.4° circular
Range Precision	0.02 m
Angular accuracy	< 0.1°
Beam divergence	0.28° (vert.) x 0.03° (horiz.)
Dimension	88 x 69 x 76 mm
Weight	760 g
Price	600 US$

Table 1: Specifications of the Livox MID-40 as given by the manufacturer

LIDAR is an instrument that was developed by Livox Technology primarily for the automotive market. Table 1 contains a few basic specifications of the sensor as provided by the manufacturer (Livox, 2019a).

The manufacturer user manual states a detection range of up to 260 metres, with some limitations on reflectivity. This puts the sensor firmly in competition with more established terrestrial LIDAR sensors and separates it from typical consumer grade 3D sensors. The manual mentions several possible application scenarios applications: “autonomous driving, robot navigation, dynamic path planning and high-precision mapping”. The sensor’s field of view is a cone with an opening angel of 38 degrees. This is reminiscent of very early terrestrial LIDAR sensors in the surveying market, often referred to as ‘window-scanners’.

Unfortunately, little is known at the time of writing about the unit’s scanning mechanism. The manufacturer’s web page specifically states that the sensor units do not ”contain any moving electronic components” (Livox Technology, 2019). However, this is not the same as claiming it to be a solid-state LIDAR. The unit’s principle is described as a non-repetitive scanning technology, where point density increases over time. The scan pattern is fixed and cannot be changed by the user.

Figure 2 exposes the scan pattern, by displaying the point cloud integrated at different points in time. This scan pattern is not consistent with a dual-axis gimbaled mirror scanner. For this we would expect a Lissajous pattern. The graphs do however resemble a rosette shaped pattern created by a Risley prism scanner (THORLABS, 2019). A Risley scanner consists of two wedge prisms that rotate around a common axis (Marshall, 1999). Depending on the ratio of the rotation speeds of the two prisms several patterns can be created which are either stable repetitive curves or non-repetitive space filling curves. It must be stated that this could only be assumed as the manufacturer does not provide detailed information about the technology that was used in the design of the scanner.

The sensor unit comes with a control software, specially designed for Livox LIDAR sensors. This software is used to record and display the real-time data within the FOV. The obtained 3D data can be exported to LAS format (“LAS Specification Version 1.3,” 2009). The Livox Viewer 0.5.0 (Livox, 2019b) as well as a Software Developer Kit (SDK) are available directly from the manufacturer.

The Livox Mid-40 series specifications were used to inform the design of the experiments described in this investigation. These values were considered for the test set-up, especially the possible ranges, precision and angular accuracy. Verifying these values was the main aim of this investigation. Establishing these characteristics helps to build a complete picture of the instrument’s behaviour as well as the limitations that the instrument could encounter in applications. Typically, specification values are obtained in controlled environments, which are difficult to recreate in a real-world situation. We attempted to repeat every measurement several times to gain better understanding of the repeatability of performance characteristics. However, not all characteristics that are relevant for a sensors suitability could be established in a quantitative manner. In part this is due to the lack of standards, but in part also due to some surprising behaviour the sensor has exhibited. We therefore split the experimental section into two, one for qualitative observations and one for quantitative assessment.

2. QUALITATIVE OBSERVTIONS

To have a first impressions of the sensor’s performance, some simple tests were performed. The aim was to capture point clouds of some typical objects and visually inspect the 3D data. Tests were performed for the long and short-range in an indoor and outdoor setting respectively. These tests comprised the behaviour of the instrument *per se* and the quality of the data that could be obtained from the scanner for the long and short-range as for an indoors and outdoors operation.

2.1 Scan Pattern

It was confirmed that the coverage of the FOV improves with time. The short-range tests show visually that the density and

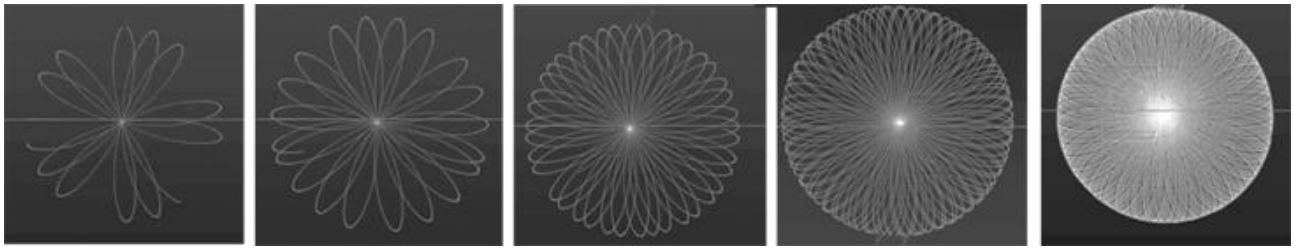


Figure 2. The sensor’s scan pattern with the point cloud becoming denser over time.

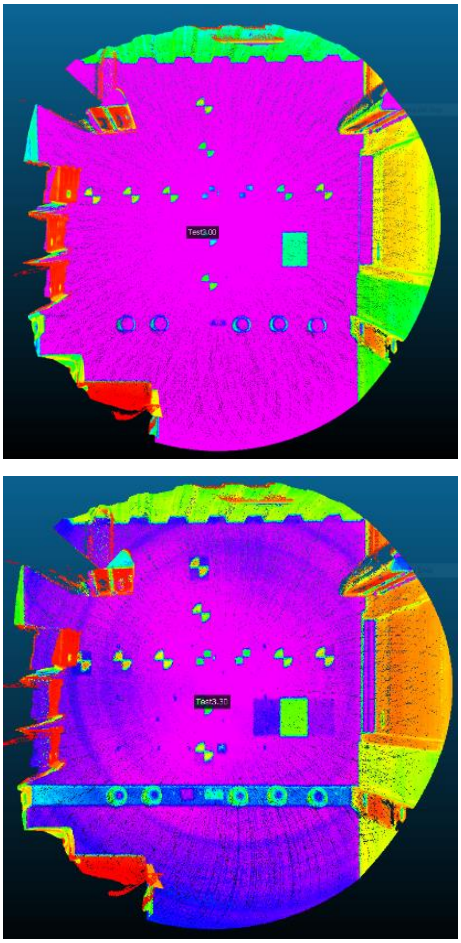


Figure 3. Point cloud captured at short-range. Data is integrated for less than minute (top) and 30 minutes (bottom).

detail of the point cloud increases with longer integration times. After 15 minutes of continuous scanning more details could be observed from the point cloud. It was only after 30 minutes of uninterrupted data collection that most details present on the test scenario were also visible on the point cloud. Figure 3 compare the detail obtained with an integration time of less than 1 minute and an integration time of 30 minutes.

This sensor does not collect points in a grid-like pattern as would be expected from typical terrestrial laser scanning systems. The rosette-like pattern causes some variations in the definition and resolution across the field of view. As the point density is not constant it is difficult to precisely identify or measure objects at a constant accuracy. This is particularly problematic for

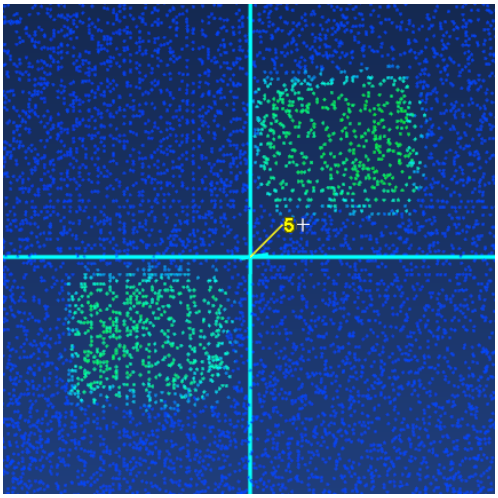


Figure 5. Centre of checker-board target with a visible gap in-between the two black squares.

measuring artificial targets which are often used in terrestrial surveying. We show two types of targets commonly used in lasers scanning: checker-board targets and spheres.

Figure 5 shows the point cloud over a standard checker-board target. It was not possible to automatically measure the target centre using available commercial software. We believe that this is in part due to the atypical scan pattern. We also observe an unusual gap in-between the two black squares that make up the pattern. Ideally, they should meet in the centre point. This leaves an additional uncertainty in manually measuring the target's centre point.

Likewise, standard commercial software was also unable to automatically identify and measure sphere targets from the sensor's point cloud. Figure 4 shows the point cloud obtained with the Livox MID-40 compared to a reference instrument the Leica RTC360. The level of noise is clearly visible as well as the 'ghost' points commonly created by mixed pixel effects.

2.2 Point Density

From the early experiments we settled to capture single scans with integration times of 5 seconds. This generates point clouds of about 300,000 points. Multiple of these single scans then can be added to create denser point clouds. In this way we create a point cloud of approximately 1,800,000 points of a facade at long range. Figure 6 shows the variation of the point density computed using Cloud Compare (Girardeau-Montaut, 2011). It is clearly visible that the density is the highest at the centre of the scan

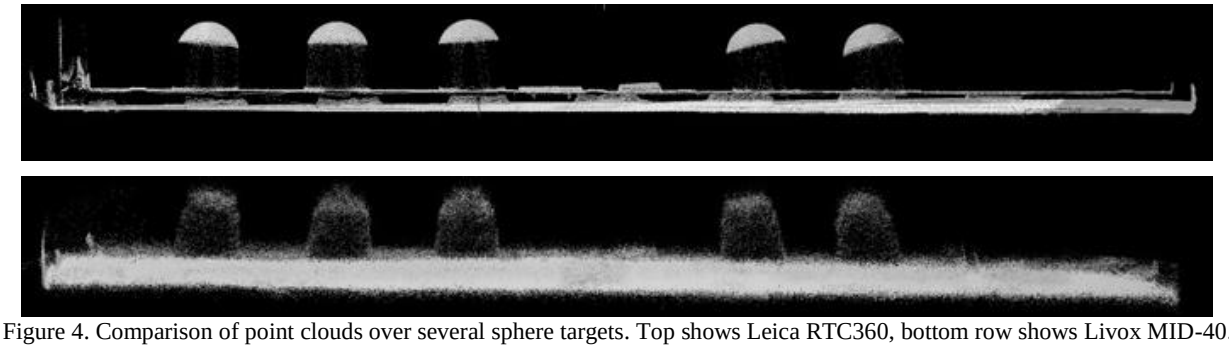


Figure 4. Comparison of point clouds over several sphere targets. Top shows Leica RTC360, bottom row shows Livox MID-40.

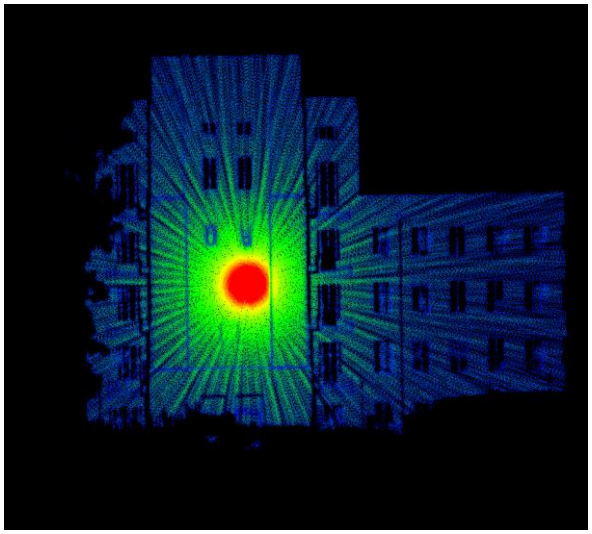


Figure 6. Point cloud density

pattern. We can also observe a star-shaped pattern as density decreases away from the centre.

2.3 Ripple Effect

Every point cloud that was obtained with the Livox MID-40 in indoors tests contains some noise artefacts which propagate along the point cloud, similar to water ripples. This effect obviously has influence on range and angular measurements. It is best visible when observing a planar object. Figure 7 shows the effect on a scan of a wall. This effect could be attributed to the potential use of a Risley prism or potential fluctuation on the laser's frequency. But not enough is known about the hardware's details to make that assessment.

2.4 Radiometric Influences

Another observation relates to the laser wavelength and the reflectivity on different colour or material. The Livox MID-40 is

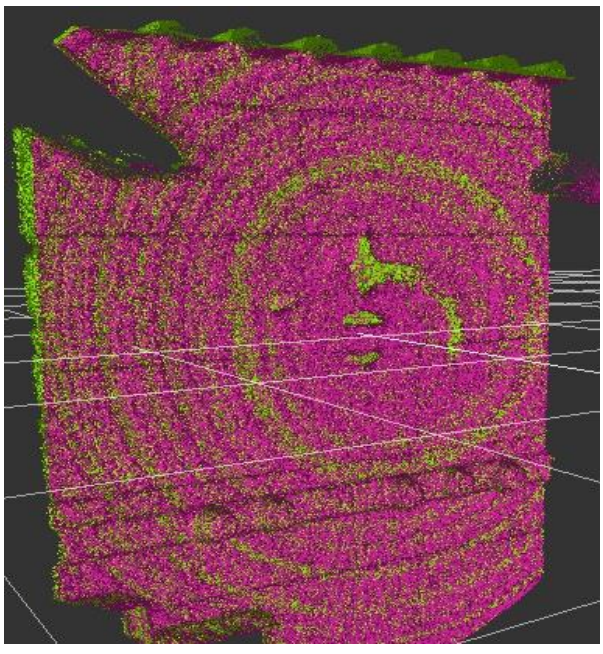


Figure 7. Ripple effect on point cloud

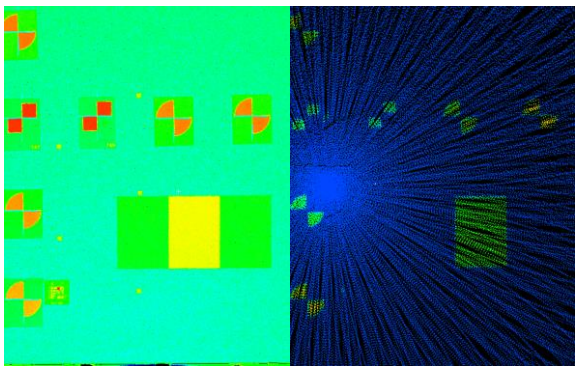


Figure 8. Leica RTC360 (left) and Livox MID-40 (right) point on different material.

specified to use a 905 nm source. Figure 8 shows a scan of three printed sheets of paper with different grey values at the centre (see also Figure 10). The point cloud shows reflected intensity as false colour for better visibility. The MID-40 cannot distinguish the brightest and the darkest pattern from the background of a white wall. However, our reference instrument the Leica RTC360, which operates at 1550 nm, can clearly separate them from the background. It is curious that after longer integration times the radiometric resolution of the instruments seems to improve (compare Figure 3).

We can observe that for the Livox MID-40 the reflected intensity has a direct influence on the range measurements. This is shown in Figure 9. While the object is a planar target, the point cloud shows a difference in depth depending on the intensity. Lighter colours are detected as 'nearer' while darker colours seem to be further away. While this effect and its causes are generally known, they have long since been eliminated in most survey-grade instruments.

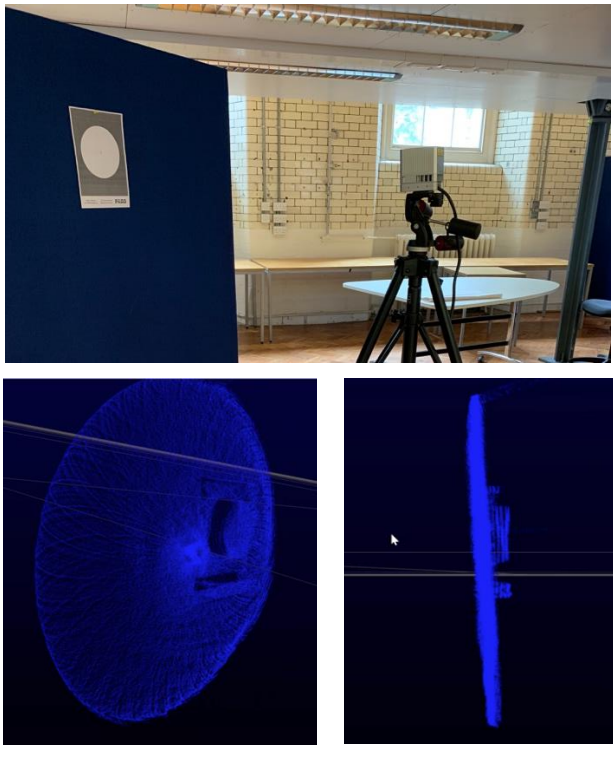


Figure 9. Scanning a planar circular target printed on paper.

3. QUANTITATIVE TEST

Given the lack of specific guidelines for verifying 3D sensors in large volumes we adapt the existing guidelines for optical 3D sensors in small volumes. Particularly we are aligning our tests to the VDI 2634 Part 2 guidelines ("VDI-Richtlinie: VDI/VDE 2634 Blatt 2 - Optische 3D-Messsysteme - Bildgebende Systeme mit flächenhafter Antastung," 2002). We have successfully relied on this guideline in previous works on sensor tests (Boehm, 2014) and its terminology is widely accepted in both the scientific community and in industry. We then conducted experiments in length measurements over a smaller indoor test field of approximately 2 meters by 1.5 meters from a distance of about 5 meters. We conducted experiments in ranging accuracy outdoors towards a flat façade at ranges from 40 to 130 meters. In the same test setup, we have also tested for planarity.

3.1 Length Measurement

The test field consists of 11 checkerboard targets arranged in the shape of a cross as shown in Figure 10. The spacing of the targets is about 30 centimetres. This arrangement allows to check 21 lengths in the horizontal direction from 0.3 meters to 2 meters and 10 lengths from 0.3 meters to 1.5 meters in the vertical direction. We repeat the experiments seven times at different times in 5-minute intervals. This is to ensure we have no thermal effects from the sensor unit heating up, which might affect performance.

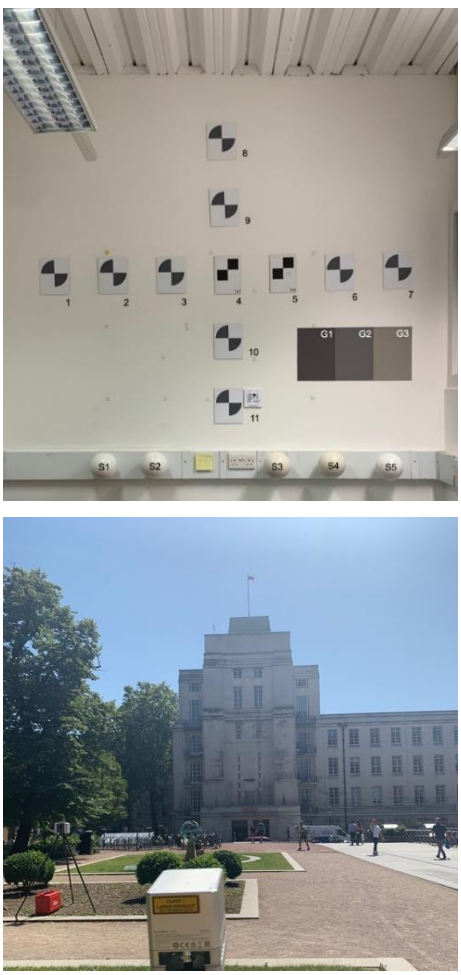


Figure 10: Test sites for the length measurement (top) and range measurement (bottom) test.

We gather reference values for the lengths with a reference instrument a Leica RTC 360 terrestrial laser scanner. From previous experience we expect the instrument to detect target centres to an accuracy of a millimetre. The deviations of the length measurements between the two instruments are plotted as a graph over the increasing length as suggested in VDI 2634. We separate plot from horizontal deviations from vertical deviations. This is to uncover any bias in the scanning mechanism. For brevity we only show 3 of the 7 experiments at 0, 5 and 30 minutes in Figure 11.

The deviations typically stay with a span of 10 millimetres and seem randomly distributed. The analysis shows no effect of the absolute length measured to the deviation from reference. This is as expected for a time-of-flight system. There is no discrepancy in horizontal deviations compared to vertical deviations. This would be in line with the assumed rotating scanning mechanism. The sensor's length measurement behaviour does not change over the observed time interval of ½ hour. This cannot always be expected for a low-cost system.

All length captured in this test are on a plane perpendicular to the sensor's optical axis. VDI 2634 would have recommended a spatial distribution of the lengths across a define measurement volume. We can use this simplified test to get an indication of the accuracy in the sensor's directional measurement. With a trigonometric approximation, neglecting any errors in ranging, we can estimate the angular accuracy from the deviation in length. With respect to the optical axis 1 mm in length measurement error corresponds to 1 hundredth of a degree in angular measurement. Likewise, 10 millimetres correspond to 1 tenth of a degree.

3.2 Range Measurement

We have conducted outdoor ranging experiments to test the sensor over longer ranges. As the laser beam is continuously spinning and its direction cannot be directly controlled it is difficult to directly assess the ranging quality. We thus try to approximate that test by assessing the ranging behaviour onto a planar object. We observe a planar sand-stone facade from distances at 40, 60, 90 and 130 meters away (see Figure 10). We fit a plane to the point cloud of the façade and calculate the distance of the sensor to the plane. The calculated range is thus not a single point range measurement by the sensor but is an estimated range from multiple measurements. This is analogous to what is sometimes described as 'modelled accuracy' for commercial terrestrial laser scanners systems.

We obtain reference values for the distances with a reference instrument, a Leica RTC360 terrestrial laser scanner. The reference scan contains both the observed plane and the Livox scanner housing. We select a centre point on the sensor's front face as the sensor's origin from which the reference distance is calculated. The treatment of the point cloud on the plane is the same as for the investigated sensor unit.

The measurement is repeated 5 times at each distance, except for the furthest station where only 4 measurements were obtained. We do this to check repeatability of the range measurements. As a plane is fit to the point cloud, we can also calculate planarity as a RMS of the point cloud at each station.

The analysis of the range measurements is not conclusive. On the one hand analysis shows deviations are in the range of ± 25 millimetre. Larger deviations seem to occur primarily at longer distances. This is supported by observing that the standard

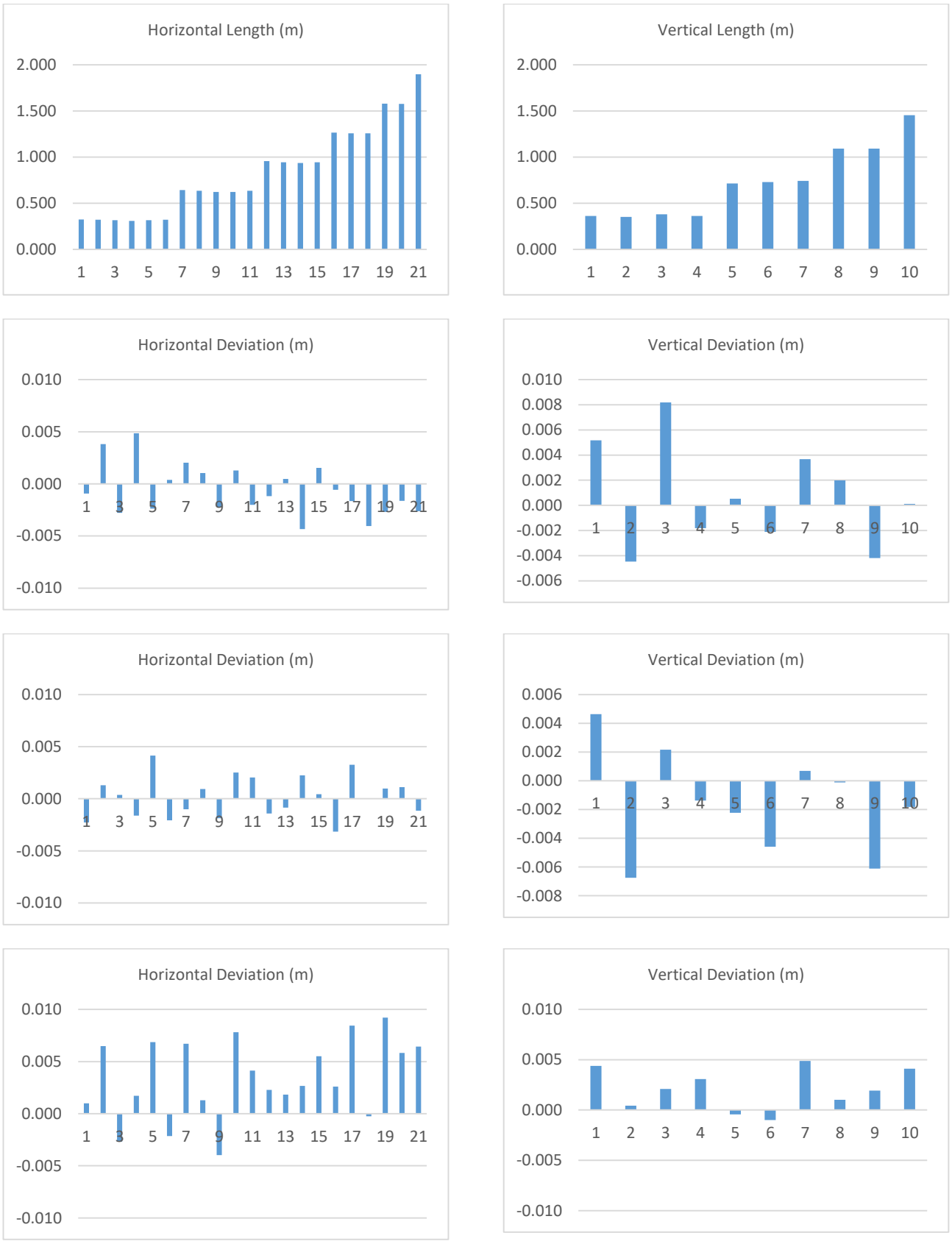


Figure 11: Length measurement experiments separated into horizontal length (first column) and vertical length (second column). The first row shows the absolute length measured. The following rows show deviations to the reference instrument at time interval 0, 5 and 30 minutes.

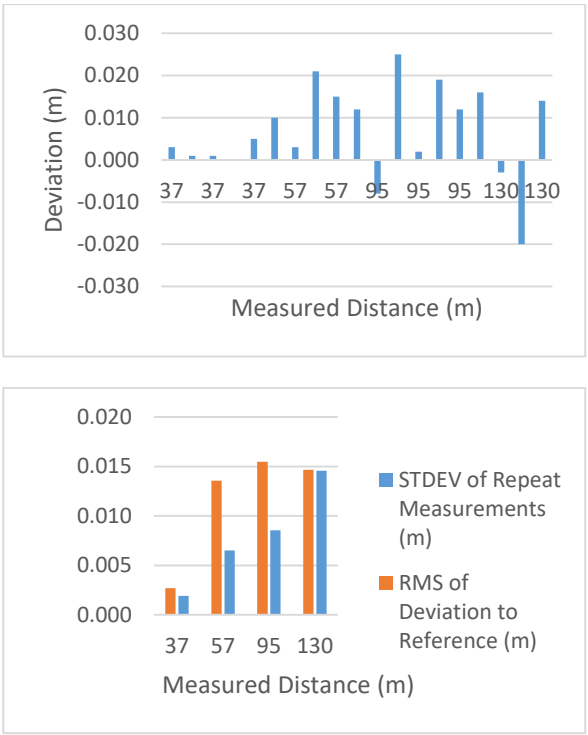


Figure 12: Deviations and repeatability for range measurements over longer distances.

deviation for the repeat measurements increases proportionally to the range. Contrary to this the RMS of the deviation to the reference for repeated measurements seems similar for all distances above 50 meters, but significantly smaller at 40 meters.so no clear dependency on distance can be found.

3.3 Flatness

The noise level of a sensor can often be assessed by checking the flatness of a point cloud over a reference plane. Obviously, at larger measurement volumes no certified test body for flatness exists. We therefore check flatness over the sand-stone façade of a building assumed to be flat. We check this with a reference instrument Leica RTC 360. The reference provides a plane fit with a standard deviation of 2 millimetres, with a maximum deviation of 5 millimetres. This is slightly worse than the instrument's specification of 1 millimetre + 10 ppm range accuracy. However, it confirms the object's flatness for the purpose of this experiment. We calculate the standard deviation of a plane-fit at the same distances as discussed above for each of the repeat measurements of the investigated sensor. Figure 13

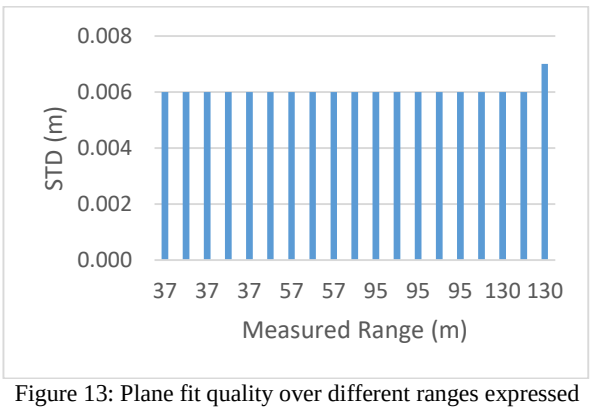


Figure 13: Plane fit quality over different ranges expressed as standard deviation of a plane fit.

shows the results obtained. The standard deviation is typically 6 millimetres. There is no visible dependency of flatness on the range.

4. CONCLUSIONS

The investigations have shown the capabilities of this very low-cost automotive LIDAR sensor. We can quantitatively characterize the sensors accuracy via the length measurement and range measurement tests. The sensor shows deviations in length measurement with a span of about 10 millimetres over short distances. This corresponds to an angular uncertainty of a tenth of a degree in the chosen setup. For close-range measurement this is clearly not a performance that would challenge existing surveying instruments. However, it is clearly within the range of specialized consumer-grade close-range sensors as described in (Boehm, 2014).

The standout feature of this sensor is its long-range capability which the manufacturer states as a maximum range of 260 meters. Our experiments show that at longer ranges the sensor unit achieves modelled distance measurements to an accuracy of ±20 millimetres. The flatness over the tested distances is constant at a standard deviation of 6 millimetres. Again, this hardly challenges established survey instruments. However, there is a clear improvement over previous generations of automotive LIDAR systems as described in (Glennie and Lichti, 2011) and (Glennie et al., 2016). In addition, in our experiments we could not observe a degradation of measurements of the time span of ½ hour.

A remaining uncertainty in our set of experiments rests with the fact that the unusual scan pattern of the sensor has not allowed us to use standard commercial software to automatically measure target centroids. Instead they had to be measured manually, which introduces additional uncertainty. With the uneven point distribution, it is impossible to estimate point density in the target areas. Visual assessment shows that point spacing around the target centres is in the order of a few millimetres. Therefore, manual target measurement clearly is a limiting factor in the length measurement experiments.

Further issues with the sensor that had an effect on the testing procedure could not be characterised quantitatively. They include the radiometric influences on range measurements. This is clearly an unwanted effect that again limits point-wise measurement accuracy. There also seems to be a systematic influence of the deflection angle which manifests itself in concentric ripples on a planar surface. This is most prominently visible at close-range. On longer ranges the general noise in the point cloud seems to hide this effect.

Within the work described here we have made no attempt to correct or calibrate for any of the effects described. At the moment there is not enough experience with the sensor to attempt this. Future work should concentrate on automating target measurement or identifying more suitable targets. This would increase confidence in the obtained values and potentially uncover further systematic patterns in the sensor's performance.

ACKNOWLEDGEMENTS

Some of the work described in this publication was carried out as part of A. Ortiz Arteaga's research project in fulfilment of the MSc degree in Geospatial Sciences at University College London.

REFERENCES

Boehm, J., 2014. Accuracy Investigation for Structured-light Based Consumer 3D Sensors. Photogramm. - Fernerkund. - Geoinformation 2014, 117–127. https://doi.org/10.1127/1432-8364/2014/0214

Girardeau-Montaut, D., 2011. CloudCompare-Open Source project. OpenSource Proj.

Glennie, C., Lichti, D.D., 2011. Temporal Stability of the Velodyne HDL-64E S2 Scanner for High Accuracy Scanning Applications. Remote Sens. 3, 539–553. https://doi.org/10.3390/rs3030539

Glennie, C.L., Kusari, A., Facchin, A., 2016. Calibration and Stability Analysis of the VLP-16 Laser Scanner. ISPRS-Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci. 55–60.

LAS Specification Version 1.3, 2009.

Ling, Z., Yuqing, M., Ruoming, S., 2008. Study on the Resolution of Laser Scanning Point Cloud, in: IGARSS 2008 - 2008 IEEE International Geoscience and Remote Sensing Symposium. IEEE, Boston, MA, USA, pp. II-1136-II–1139. https://doi.org/10.1109/IGARSS.2008.4779200

Livox, 2019a. Livox Mid Series User Manual v1.2.

Livox, 2019b. Livox Viewer 0.5.0 [WWW Document]. URL https://www.livoxtech.com/hub/downloads

Livox Technology, 2019. Mid-40 lidar sensor - Livox [WWW Document]. URL https://www.livoxtech.com/mid-40-and-mid-100 (accessed 10.30.19).

Marshall, G.F., 1999. Risley prism scan patterns, in: Beiser, L., Sagan, S.F., Marshall, G.F. (Eds.), . Presented at the SPIE's International Symposium on Optical Science, Engineering, and Instrumentation, Denver, CO, pp. 74–86. https://doi.org/10.1117/12.351658

Moras, J., Rodriguez, F.S.A., Drevelle, V., Dherbomez, G., Cherfaoui, V., Bonnifait, P., 2012. Drivable space characterization using automotive lidar and georeferenced map information, in: 2012 IEEE Intelligent Vehicles Symposium. Presented at the 2012 IEEE Intelligent Vehicles Symposium (IV), IEEE, Alcal de Henares , Madrid, Spain, pp. 778–783. https://doi.org/10.1109/IVS.2012.6232252

Royal Institution of Chartered Surveyors, 2014. Measured surveys of land, buildings and utilities: RICS guidance note, global.

Thomson, C., Boehm, J., 2014. Indoor Modelling Benchmark for 3D Geometry Extraction. ISPRS - Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci. XL–5, 581–587. https://doi.org/10.5194/isprsarchives-XL-5-581-2014

THORLABS, 2019. Risley Prism Scanner [WWW Document]. URL https://www.thorlabs.com/images/tabimages/Risley_Prism_Scanner_App_Note.pdf

VDI-Richtlinie: VDI/VDE 2634 Blatt 2 - Optische 3D-Messsysteme - Bildgebende Systeme mit flächenhafter Antastung, 2002.

Loam_livox: A fast, robust, high-precision LiDAR odometry and mapping package for LiDARs of small FoV

Jiarong Lin and Fu Zhang

***Abstract*— LiDAR odometry and mapping (LOAM) has been playing an important role in autonomous vehicles, due to its ability to simultaneously localize the robot's pose and build high-precision, high-resolution maps of the surrounding environment. This enables autonomous navigation and safe path planning of autonomous vehicles. In this paper, we present a robust, real-time LOAM algorithm for LiDARs with small FoV and irregular samplings. By taking effort on both front-end and back-end, we address several fundamental challenges arising from such LiDARs, and achieve better performance in both precision and efficiency compared to existing baselines. To share our findings and to make contributions to the community, we open source our codes on Github[1].**

I. Introduction

With the ability to provide long range, highly accurate 3D measurements of the surrounding environment, light detection and ranging (LiDARs) is becoming an essential sensor in many robotic applications, such as autonomous driving vehicles [1], drones [2, 3], surveying, and mapping [4, 5]. To enable massive use in these areas, recent developments in LiDAR technologies have been focusing on lowering the device cost while increasing its reliability [6]. In this trend, one class of LiDARs that gains increasingly interests and developments are solid state LiDARs, which come with various implementations, such as micro-electro-mechanical-system (MEMS) scanning, optical phase array (OPA), Risley prism, etc. Being massively produced [2], these high performance and extremely low-cost LiDARs hold the potential to promote or radically change the robotics industry.

Despite their superiority in cost, reliability, and possibly performance against the conventional mechanical spinning LiDARs, such as Velodyne Puck [3], solid state LiDARs have many new features that bring significant challenges to the LiDAR navigation and mapping. These features are (to explain these features, we take the Livox MID40 LiDAR [2] as an example due to its wide availability):

Small FoV: solid state LiDARs usually have very small field of view (FoV). For examples, Livox MID40 has a front facing, conical shaped FoV spanning 38.4 degrees. Other solid state LiDARs such MEMS LiDARs also suffer from similar small FoV problem due to the large size of the MEMS mirror preventing large steering angles. Comparing to conventional spinning LiDAR (see Fig. 3), the reduced FoV will lead to very fewer features in a frame, making the subsequent feature matching prone to degenerate and easily disturbed by moving objects. Although a larger FoV can be obtained by combining multiple LiDARs, it considerably increases the sensor cost and weight.

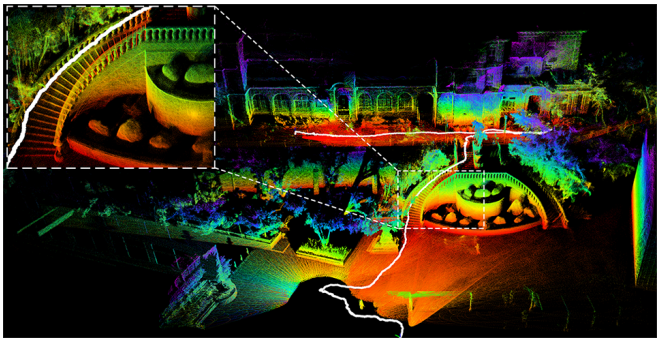


Fig. 1: The 3D map of the Chong Yuet Ming Cultural Center in the University of Hong Kong (HKU).

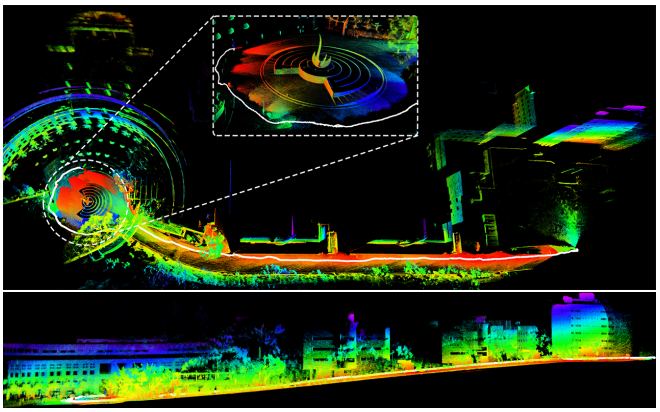


Fig. 2: The large scale mapping of the Hong Kong University of Science and Technology (HKUST) main campus, the upper and lower images are the bird-view and side-view, respectively. In above images, the white path is the trajectory of the LiDAR, points are colored by their heights.

Irregular scanning pattern: existing spinning LiDARs have multiple laser-receiver pairs stacking in a vertical row. Rotating all pairs as a whole leads to a collection of parallel rings. This regular scanning greatly simplifies the feature extraction. For example, a corner is easily computed by differentiating the depth of points on the line. In constrat, the scanning pattern of solid state LiDARs is quite irregular. For example, the Livox MID40 has a rosette-like scanning pattern (see Fig. 4) where two neighboring scanning petals are separated far apart.

Non-repetitive scanning: to maximize the coverage ratio even when the LiDAR is static, non-repetitive scanning is

J. Lin and F. Zhang are with the Department of Mechanical Engineering, Hong Kong University, Hong Kong SAR., China. {jiarong.lin, fuzhang}@hku.hk

[1] https://github.com/hku-mars/loam_livox

[2] https://www.livoxtech.com/mid-40-and-mid-100

[3] https://velodynelidar.com/vlp-16.html

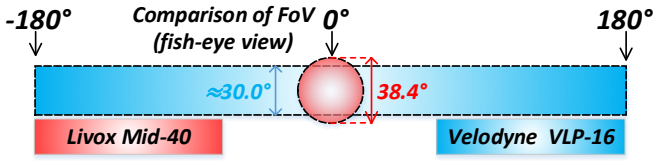


Fig. 3: FoV of Livox Mid-40 and Velodyne PUCK (VLP-16).

usually adopted [7] where the scanning trajectory never repeats itself (see Fig. 4).

Motion blur: due to the continuous scanning of a single laser head, the 3D points measured in one frame are really sampled at different times as the LiDAR is continuously moving. The in-frame motion will distort the point clouds and cause motion blur.

To address the problems mentioned above, we develop a software package named "Loam_Livox", which addresses many key issues including feature extraction and selection in a very limited FoV, robust outliers rejection, moving objects filtering and motion distortion compensation. Without other sensors such as IMU, GPS, and cameras, our algorithm calculates the LiDAR poses in real time (i.e. odometry) by registering its point cloud to a specified range of local map. Some of the results we obtained are shown in Fig. 1 and Fig. 2, where we can tell the precision of the algorithm from the level of details of the stairs and railing (Fig. 1), as well as versatility for large-scale mapping (Fig. 2).

II. Related work

State estimation and map-building are the fundamental prerequisites for intelligent robots. In the past recent years, we have seen great efforts being made in the field of simultaneous localization and mapping (SLAM), including both vision-based and laser-based approaches. In this paper, we mainly focus on the problem of laser-based SLAM.

Besl *et al* [8] first proposed the Iterative Closest Points (ICP) method for scan matching, which builds the basic operation for odometry. Building on this, Mendes *et al* [9] proposed a pose-graph SLAM to correct the drift in sequential scan matching, and demonstrated its effectiveness in a high definition LiDARs, Velodyne HDL 64.

While the ICP algorithm performs well for 3D scans with dense points, its effectiveness considerably degrades when the points in a scan are sparse where the two scans do not scan the same location on an object. To solve this problem, Pulli *et al* [10] proposed a point-to-plane error metric. This metric is used together with the point-to-point metric in [8] and called the generalized ICP in [11]. Zhang *et al* [12] and Shan *et al* [13] also used the point-to-edge metric in the context of LiDAR odometry and mapping.

Besides the geometry features mentioned above, 3D keypoints based method [14]–[16] have also been proposed. These methods required less computation resources, by extracting keypoints from dense point cloud with detector like Point Feature Histograms (PFH) [14, 15], Viewpoint Feature Histograms (VFH) [16], etc. Considering the point cloud characteristic of our scenarios and the demands of real-time performance, we use point-to-edge and point-to-plane feature in our work, inspired by the work of [12, 13].

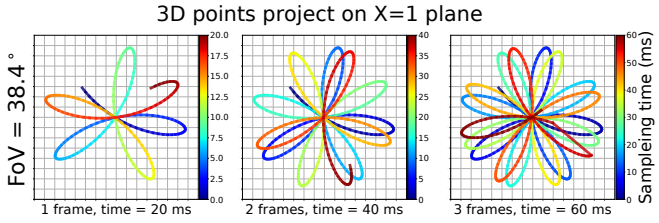


Fig. 4: The scanning trajectory of 3D points projected on the plane of $1m$ distance in front, where the color encodes the sampling time.

To eliminate the effects of motion blur caused by LiDAR movement, authors in [12] , [17] and [18] compensate the movements in front-end processing by linear interpolating the LiDAR pose. More recently, Gentil *et al* [19] formulates an optimization problem in the back-end processing to compensate the LiDAR movement. Compared to the previous work, the back-end processing method achieves better performance but cannot run in real-time.

While most of the previous work were based on spinning LiDARs, in this work, we focus on the odometry and mapping with solid-state LiDARs of small FOVs. Our contributions are: (1) we develop a complete LOAM algorithm for LiDARs with small FOVs. The algorithms is carefully engineered and made open source to benefit the community; (2) we increase the accuracy and robustness of the LOAM algorithm by considering the ow-level physical properties of LiDAR sensors in the front-end processing; (3) we propose a simple yet effective motion compensation method, the piecewise processing, and parallelize its implementation. Experiments show that the piecewise processing outperforms linear interpolation in terms of accuracy and running efficiency.

III. Points selection and feature extraction

The overview or our system is shown as Fig. 5, whose front-end processing comprises of the point selection and feature extraction. Considering the measuring mechanism of a LiDAR sensor its low-level physical properties (e.g., laser spot size, signal noise ratio), we perform a point level selection to extract the "good points".

A. Points selection

We compute the following features of each 3D point $\mathbf{P} = [x, y, z]$, where the $X-Y-Z$ axis correspond to the Front-Left-Up (FLU) of a LiDAR (see Fig. 6 (a)).

Depth D is the distance of the measured point to the LiDAR sensor.

$$D(\mathbf{P}) = \sqrt{x^2 + y^2 + z^2} \tag{1}$$

The laser deflection angle ϕ is the angle between X axis and laser ray

$$\phi(\mathbf{P}) = \tan^{-1}\left(\sqrt{(y^2 + z^2)/x^2}\right) \tag{2}$$

The intensity I is

$$I(\mathbf{P}) = R/D(\mathbf{P})^2 \tag{3}$$

where R is the object reflectivity, measured by the LiDAR sensor (some LiDARs, e.g., Velodyne Puck, returns the

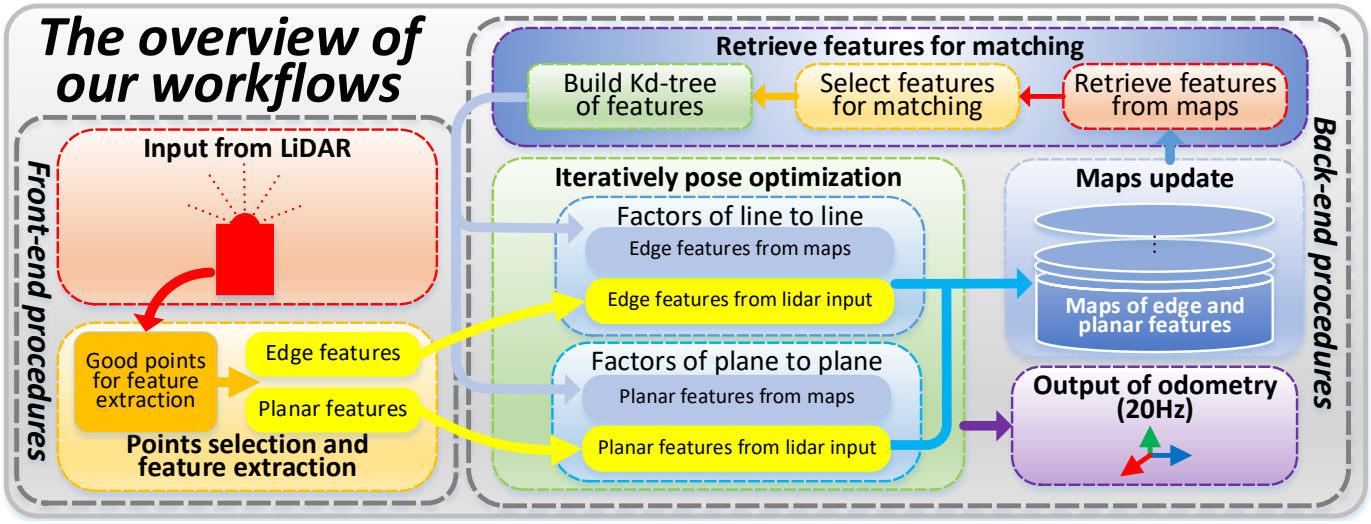


Fig. 5: The overview of our workflows. Each new frame is matched directly with the global map to provide the odometry output. The matching result is in turn used to register the frame to the global map, leading to the same rate (i.e., 20 Hz) of odometry output and map update. In our implementation, only the feature points (i.e., edge points and plane points) are saved in memory and all the raw points are saved in hard disk for possible offline processing (e.g., offline global optimization).

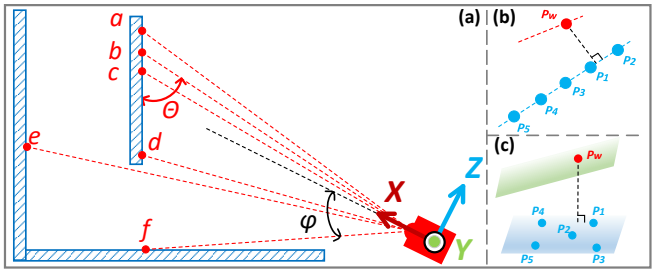


Fig. 6: (a) Illustration of incident angle θ, deflection angle ϕ; (b) residual of edge-to-edge; (c) residual of plane-to-plane.

intensity instead of reflectivity. In this case, the intensity is directly available). Small intensity $I(\mathbf{P})$ means the point is either far from the LiDAR sensor (large $D(\mathbf{P})$) or the object reflectivity R is low.

The incident angle θ is the angle between the laser lay and the local plane around the measured point (Fig. 6 (a)).

$$\theta(\mathbf{P}_b) = \cos^{-1}\left(\frac{(\mathbf{P}_a - \mathbf{P}_c)\cdot \mathbf{P}_b}{|\mathbf{P}_a - \mathbf{P}_c|\,|\mathbf{P}_b|}\right) \tag{4}$$

To increase the localization and mapping accuracy, we remove any of the following points:

1. Points nearing to the fringe of the FoV. (e.g., $\phi(\mathbf{P}) \geq 17°$ for Livox MID40). In such area, scanning trajectory has large curvatures, leading to the feature extraction in Section III.B less reliable.
2. Points with too large or too small intensity (e.g. $I(\mathbf{P}) \leq 7 \times 10^{-3}$,or $I(\mathbf{P}) \geq 1 \times 10^{-1}$ for MID40). This is because intensity directly indicates the strength of the received laser signal. Too large intensity (signal) usually leads to saturation or distortion in the receiving circuitry and decreases the ranging accuracy. On the other hand, too small intensity (signal) usually leads to lower signal noise ratio, which also deteriorate the ranging accuracy.
3. Points with incident angles near to π or 0 (e.g. $\theta(\mathbf{P}) \leq 5°$,or $\theta(\mathbf{P}) \geq 175°$ for MID40), like point $\mathbf{P}_f$ in Fig. 6 (a). This is because the laser spot caused by the nonzero divergence angle of the laser beam will be considerably elongated. As a result, the measured range is the average of the area covered by the large spot instead of a specific point.
4. Points hidden behind an objects (e.g., $\mathbf{P}_e$ in Fig. 6 (a)), which will cause a false edge feature otherwise. A point $\mathbf{P}_e$ is a hidden point if:

$$|\mathbf{P}_e - \mathbf{P}_d| \geq 0.1|\mathbf{P}_e|, \text{ and} |\mathbf{P}_e| > |\mathbf{P}_d|$$

where $\mathbf{P}_d$ is the nearest measurement point in scanning order.

B. Feature extraction

After points selection, we perform feature extraction to extract features from the "good points". We extract plane and edge features by computing the local smoothness of the point candidate as in [12]. Furthermore, to mitigate the matching degeneration due to the small number of features caused by the limited FoV and the point selection, we employ the LiDAR reflectivity as the 4-th dimensional measurement. If the reflectivity of a 3D point is considerably different its neighborhood points, we treat it as a point of edge feature (edge in the reflectivity due to materials change, in contrast to the edge in geometry due to shape change). Such points are beneficial in some of the degeneration cases like facing a wall with closed doors and windows.

IV. Iterative Pose Optimization

Due to the non-repetitive scanning mentioned in Section. I, the extracted feature cannot be constantly tracked like in [12, 13, 19]. A simple example is that, even when the LiDAR is static, the scanned trajectory (and feature points) are different from the previous frame. In our work, we use an iterative pose optimization procedure to calculate the LiDAR pose. With the proper implementation detailed later, we achieve real-time odometry and mapping, both at 20Hz.

A. Residual of edge-to-edge

Denote $\boldsymbol{\mathcal{E}}_k$ and $\boldsymbol{\mathcal{E}}_m$ the set of all edge features (see Section. III-B) in the current frame and in the map, respectively. For each point in $\boldsymbol{\mathcal{E}}_k$, we find 5 nearest points from $\boldsymbol{\mathcal{E}}_m$ (see Fig. 6 (b)). To boost the searching speed, we build a *KD-tree* of $\boldsymbol{\mathcal{E}}_m$ (see Fig. 5). Moreover, the KD-tree is built by another parallel thread once the last registered frame/sub-frame is received (see Fig. 7). This makes the KD-tree immediately available when the new frame is received.

Let $\mathbf{P}_l$ be a point in $\boldsymbol{\mathcal{E}}_k$ of the current frame (k-th frame). Noticing that the point $\mathbf{P}_l$ in $\boldsymbol{\mathcal{E}}_k$ is in the local LiDAR frame while points of $\boldsymbol{\mathcal{E}}_m$ are registered in the global map, to find the nearest points of $\mathbf{P}_l$ in $\boldsymbol{\mathcal{E}}_m$, we need to project it to the global map by the following transformation.

$$\mathbf{P}_w = \mathbf{R}_k \mathbf{P}_l + \mathbf{T}_k \tag{5}$$

where $(\mathbf{R}_k, \mathbf{T}_k)$ is the LiDAR pose when the last point in current frame is sampled, and needs to be determined by the pose optimization. Here we use the LiDAR pose at the last point in a frame to represent the pose of the whole frame, and all points in that frame are projected to the global map using this pose. Also notice that the last point in the current frame is essentially the first point in the next frame.

Let $\mathbf{P}_i$ denote the i-th nearest points of $\mathbf{P}_w$ of $\boldsymbol{\mathcal{E}}_m$. To make sure that $\mathbf{P}_i$ is indeed on a line, we compute the mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ formed by the m nearest points of $\mathbf{P}_w$. We set m to 5 in our work. If the biggest of eigenvalue of $\boldsymbol{\Sigma}$ is three times larger than the second biggest eigenvalue, we assure that the nearest points of $\mathbf{P}_w$ form a line on which $\mathbf{P}_w$ should lie. The residual is then computed as (Fig. 6 (b)).

$$\mathbf{r}_{e2e} = \frac{|(\mathbf{P}_w - \mathbf{P}_5) \times (\mathbf{P}_w - \mathbf{P}_1)|}{|\mathbf{P}_5 - \mathbf{P}_1|} \tag{6}$$

B. Residual of plane-to-plane

Similar to the edge feature points, for a point in the planar feature set $\boldsymbol{\mathcal{P}}_k$ of current frame, we find 5 nearest points in the planar feature set $\boldsymbol{\mathcal{P}}_m$ of the map (see Fig. 6(c)). We also assure these 5 nearest points are indeed within the same plane by computing their covariance matrix Σ. If the smallest eigenvalue of $\boldsymbol{\Sigma}$ is three times less than the second smallest eigenvalue, we compute the distance of the plane point in the current frame to the plane formed by the 5 points in the same plane, as follows, and add this residual to pose optimization.

$$\mathbf{r}_{p2p} = \frac{(\mathbf{P}_w - \mathbf{P}_1)^T ((\mathbf{P}_3 - \mathbf{P}_5) \times (\mathbf{P}_3 - \mathbf{P}_1))}{|(\mathbf{P}_3 - \mathbf{P}_5) \times (\mathbf{P}_3 - \mathbf{P}_1)|} \tag{7}$$

C. In-frame motion compensation

As mentioned previously, the 3D points are sampled at different time of different poses (i.e., motor blur) as the LiDAR motion is continuously undergoing. To eliminate the effect of motion blur, we propose two methods as follows:

1) Piecewise processing: A simple yet effective way to eliminate the effect of motion blur is piecewise processing. We divide an incoming frame into three sequential sub-frames. Then match these three sub-frames to the same map accumulated so far independently. During the scan matching

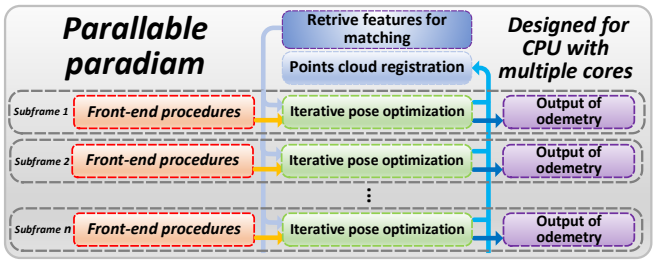


Fig. 7: Our parallel paradigm for CPU with multiple cores. Each sub-frame is matched with the global map independently on a dedicated thread. The matched sub-frame is then registered to the global map and become a part of the map. Another dedicated thread receives the new registered sub-frame and build a KD three of the updated map to be used in the next frame.

of each sub-frame, all its points are projected to the global map using the LiDAR pose at the end point of that sub-frame. By doing so, the time interval of each sub-frame is 1/3 of the original frame. Although this method seems very simple, it works surprisingly well as shown in the results. Additionally, this piecewise processing has the benefits of utilizing the multi-core structure in modern CPUs by parallelizing the matching of each sub-frame (see Fig. 7).

2) Linear interpolation: Another commonly used motion compensation method is linear interpolation, as in [12, 19]. Denote $(\mathbf{R}_k, \mathbf{T}_k)$ the LiDAR pose at the last point of the current frame and $(\mathbf{R}_{k-1}, \mathbf{T}_{k-1})$ the previous frame, $(\mathbf{R}_{k-1}^k, \mathbf{T}_{k-1}^k)$ the relative rotation and translation between the previous and the current frame, then:

$$\mathbf{R}_k = \mathbf{R}_{k-1}\mathbf{R}_{k-1}^k, \quad \mathbf{T}_k = \mathbf{R}_{k-1}\mathbf{T}_{k-1}^k + \mathbf{T}_{k-1}$$

Assume t_{k-1} is the sampling time of the last point in the previous frame. For any point sampled at time t in the current frame, we have $t \in [t_{k-1}, t_k]$. Compute $s = (t - t_{k-1})/(t_k - t_{k-1})$, then, the linearly interpolated pose at time t is:

$$\mathbf{R}_{k-1}^t = e^{\hat{\boldsymbol{\omega}}\theta s}, \quad \mathbf{T}_{k-1}^t = s\mathbf{T}_{k-1}^k$$

where θ is the magnitude and $\boldsymbol{\omega}$ is the unit vector of of the rotation axis of $\mathbf{R}_{k-1}^k$, respectively. $\hat{\boldsymbol{\omega}}$ is the skew symmetric matrix of $\boldsymbol{\omega}$. From the Rodrigue's formula [20], we have:

$$\mathbf{R}_{k-1}^t = \mathbf{I} + \hat{\boldsymbol{\omega}}\sin(s\theta) + \hat{\boldsymbol{\omega}}^2(1 - \cos(s\theta))$$

which implies that only $\sin(s\theta)$ and $\cos(s\theta)$ needs to be computed for each point of the current frame, while the rests remain constant. This saves some computations. With $\mathbf{R}_{k-1}^t$, the LiDAR pose at the current time is:

$$\mathbf{R}_t = \mathbf{R}_{k-1}\mathbf{R}_{k-1}^t, \quad \mathbf{T}_t = \mathbf{R}_{k-1}\mathbf{T}_{k-1}^t + \mathbf{T}_{k-1} \tag{8}$$

Then we can project the point at time t to the global map by the interpolated pose, as follows:

$$\mathbf{P}_w(t) = \mathbf{R}_t\mathbf{P}_l + \mathbf{T}_t \tag{9}$$

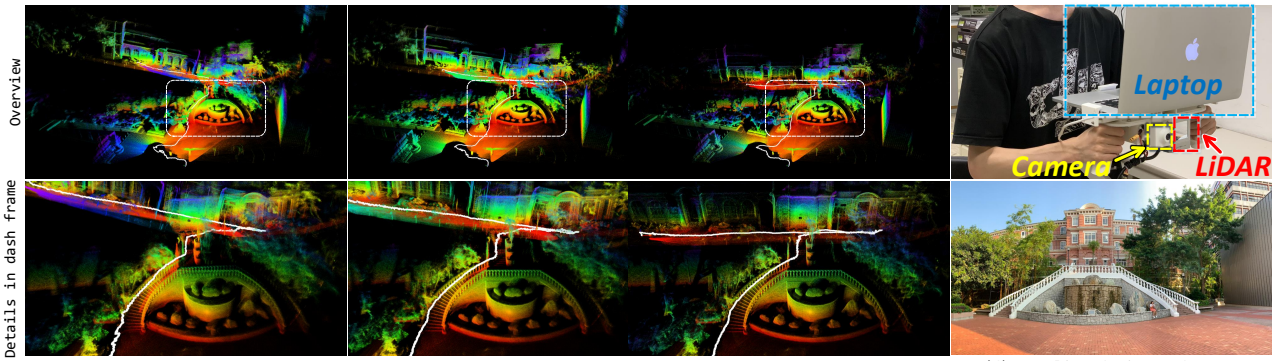


Fig. 8: The comparison of different motion compensation methods. The first column shows the results without any motion compensation, the second is with linear interpolation, and the third column is piecewise processing. The upper picture in the fourth column is our hand-held device for data collection, the lower picture is the RGB image of the mapped area.

Algorithm 1: Iterative LiDAR pose optimization

Input : The edge set $\mathcal{E}_k$ and plane set $\mathcal{P}_k$ from the current (sub-) frame; The edge set $\mathcal{E}_m$ and plane set $\mathcal{P}_m$ from maps; The LiDAR pose of the previous frame $(\mathbf{R}_{k-1}, \mathbf{T}_{k-1})$.
Output: The pose of the current frame $(\mathbf{R}_k, \mathbf{T}_k)$.
Start : $\mathbf{R}_k \leftarrow \mathbf{R}_{k-1}$, $\mathbf{T}_k \leftarrow \mathbf{T}_{k-1}$
for *Iterative pose optimization is not converged* **do**
  for $\mathbf{P}_l \in \mathcal{E}_k$ **do**
    Compute $\mathbf{P}_w$ via (5) (or (9)).
    Find 5 nearest points $\{\mathbf{P}_{1\sim5}\}$ of $\mathbf{P}_w$ in $\mathcal{E}_m$.
    if $\{\mathbf{P}_{1-5}\}$ *are indeed in a line* **then**
      Add edge-to-edge residual $\mathbf{r}_{e2e}$ via (6).
  for $\mathbf{P}_l \in \mathcal{P}_k$ **do**
    Compute $\mathbf{P}_w$ via (5) (or (9)).
    Find 5 nearest points $\{\mathbf{P}_{1\sim5}\}$ of $\mathbf{P}_w$ in $\mathcal{P}_m$.
    if $\{\mathbf{P}_{1\sim5}\}$ *are indeed a plane* **then**
      Add plane-to-plane residual $\mathbf{r}_{p2p}$ via (7).
  Perform pose optimization with 2 iterations.
  Recompute $\mathbf{r}_{e2e}$ and $\mathbf{r}_{p2p}$, then remove 20% of the biggest residual.
  for *a maximal number of iterations* **do**
    if *the nonlinear optimization converges* **then**
      Break;

D. Outliers rejection, dynamic objects filtering

To avoid moving object in real environments bringing down the accuracy of scan matching, we perform a dynamic objects filtering as follows: in each iteration of the iterative pose optimization, we refind the nearest neighbors of each feature point and add the edge-to-edge residual (6) and plane-to-plane residual (7) to the objective function, we first perform pose optimization with a small number of iterations (e.g., 2 used in our experiments). Using the optimization results, we compute the two residuals in (6) and (7), and remove the first 20% largest residuals. With the outliers removed, a full pose optimization is finally performed. The complete iterative pose optimization algorithm is summarized in Algorithm. 1.

V. Results

A. Evaluation of mapping

The comparisons of the two motion compensation methods of are shown in Fig. 8, where we can see that, without any motion compensation (Fig. 8 (a)), the mapping is very blurry in local areas (e.g., stairs, railing) and distorted in larger scale (e.g., the building is curved). In contrast, with the motion compensation, both of the linear interpolation and piecewise processing effectively eliminate the motion blur, and the stair steps and railing are distinguishable one from another. However, the linear interpolation has a considerable long-term drift, as seen by the curved building in the upper figure of Fig. 8 (b). This is because the data are collected by hand-held devices and the movement could be quite jerky and cannot be accurately captured by simple linear interpolation.

B. Evaluation of odometry

We evaluate the localization of our algorithm by comparing with the measurement of GPS, shown in Fig. 9. We compute the distance of two positions of our odometry and then compare it with the measurement of GPS. The results on two datasets are 0.41% and 0.65%, respectively, implying that the localization is of high accuracy.

Furthermore, we evaluate the accuracy of rotation by comparing our result with the motion capture system (mocap) shown in Fig. 10). The results show that the trajectories of our odometry and mocap are very close and the average error of Euler angles in all three directions is as low as $1.1°$.

C. Evaluation of running performance

We evaluate the time consumption of our algorithm and the current baseline [4] (both algorithm eliminates the motion blur with piecewise processing) on two platforms, the desktop PC (with *i7-9700K*) and onboard-computer (DJI manifold2

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

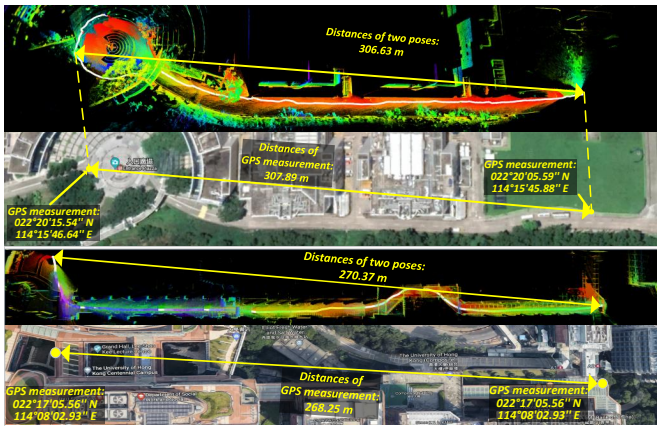


Fig. 9: The localization accuracy on two datasets: outdoor (upper) and indoor (lower). In each dataset, we compare our odometry results with Google maps and compute the traveled distance.

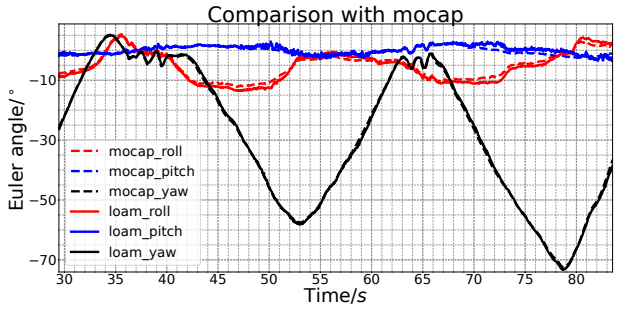


Fig. 10: The comparison between our results and motion capture (mocap) system, the dashed line is measured by mocap, and the solid line is the odometry output from our algorithm.

[5] with *i7-8550U*). As shown in Table. I, benefiting from the parallelization among sub-frame registration, as well as between feature matching and KD-tree building, our algorithms run *2∼3* times faster than the baseline.

D. Others

Due to the space limit, we strongly recommend the reader to review our code [6] and more results contained there [7].

VI. Conclusion and Discussion

This paper presented an odometry and mapping algorithm for LiDARs with small FOVs. The algorithm inherits the basic structure and techniques (e.g., feature extraction, mating, motion compensation by linear interpolation) of standard LOAM algorithm, but with several key new contributions, such as point selection, iterative pose optimization, and implementation parallelization. The developed algorithm has its odometry and mapping both running in real time (i.e., 20 Hz). While achieving a high level of accuracy in mapping and localization, the sequential scan matching is inherently drifting. Reducing this drift by using techniques like loop closure and sliding window optimization will be further researched in the future.

[5] https://www.dji.com/cn/manifold-2
[6] https://github.com/hku-mars/loam_livox
[7] https://github.com/ziv-lin/loam_livox_paper_res

	Desktop PC @4.0∼4.8 Ghz	Desktop PC parallel	Onboard PC @3.0∼3.5 Ghz	Onboard PC parallel
Ours	35.68 ms	17.24 ms	54.60 ms	32.54 ms
Baseline	109.00 ms	NaN	125.13 ms	NaN

TABLE I: The time consumption for each frame of our algorithm and the baseline[4], where "Desktop PC parallel" and "Onboard PC parallel" use 3 threads for point cloud registration.

References

[1] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt, *et al.*, "Towards fully autonomous driving: Systems and algorithms," in *2011 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2011, pp. 163–168.

[2] A. Bry, A. Bachrach, and N. Roy, "State estimation for aggressive flight in gps-denied environments using onboard sensing," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 1–8.

[3] F. Gao, W. Wu, W. Gao, and S. Shen, "Flying on point clouds: Online trajectory generation and autonomous navigation for quadrotors in cluttered environments," *Journal of Field Robotics*, vol. 36, no. 4, pp. 710–733, 2019.

[4] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann, "6d slam3d mapping outdoor environments," *Journal of Field Robotics*, vol. 24, no. 8-9, pp. 699–722, 2007.

[5] B. Schwarz, "Lidar: Mapping the world in 3d," *Nature Photonics*, vol. 4, no. 7, p. 429, 2010.

[6] "Ces 2018: Waiting for the $100 lidar." [Online]. Available: https://spectrum.ieee.org/cars-that-think/transportation/sensors/ces-2018-how-a-new-generation-lidars-is-redefining-the-car

[7] "Point cloud characteristics of livox-lidar." [Online]. Available: https://www.livoxtech.com/3296f540ecf5458a8829e01cf429798e/downloads/Pointcloudcharacteristics.pdf

[8] P. J. Besl and N. D. McKay, "Method for registration of 3-d shapes," in *Sensor fusion IV: control paradigms and data structures*, vol. 1611. International Society for Optics and Photonics, 1992, pp. 586–606.

[9] E. Mendes, P. Koch, and S. Lacroix, "Icp-based pose-graph slam," in *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE, 2016, pp. 195–200.

[10] K. Pulli, "Multiview registration for large data sets," in *Second International Conference on 3-D Digital Imaging and Modeling (Cat. No. PR00062)*. IEEE, 1999, pp. 160–168.

[11] A. Segal, D. Haehnel, and S. Thrun, "Generalized-icp." in *Robotics: science and systems*, vol. 2, no. 4. Seattle, WA, 2009, p. 435.

[12] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time." in *Robotics: Science and Systems*, vol. 2, 2014, p. 9.

[13] T. Shan and B. Englot, "Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4758–4765.

[14] R. B. Rusu, Z. C. Marton, N. Blodow, and M. Beetz, "Learning informative point classes for the acquisition of object model maps," in *2008 10th International Conference on Control, Automation, Robotics and Vision*. IEEE, 2008, pp. 643–650.

[15] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (fpfh) for 3d registration," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 3212–3217.

[16] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu, "Fast 3d recognition and pose using the viewpoint feature histogram," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 2155–2162.

[17] S. Hong, H. Ko, and J. Kim, "Vicp: Velocity updating iterative closest point algorithm," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 1893–1898.

[18] D. Droeschel and S. Behnke, "Efficient continuous-time slam for 3d lidar-based online mapping," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–9.

[19] C. L. Gentil, T. Vidal-Calleja, and S. Huang, "In2laama: Inertial lidar localisation autocalibration and mapping," *arXiv preprint arXiv:1905.09517*, 2019.

[20] R. M. Murray, *A mathematical introduction to robotic manipulation*. CRC press, 2017.

A fast, complete, point cloud based loop closure for LiDAR odometry and mapping

Jiarong Lin and Fu Zhang

***Abstract*— This paper presents a loop closure method to correct the long-term drift in LiDAR odometry and mapping (LOAM). Our proposed method computes the 2D histogram of keyframes, a local map patch, and uses the normalized cross-correlation of the 2D histograms as the similarity metric between the current keyframe and those in the map. We show that this method is fast, invariant to rotation, and produces reliable and accurate loop detection. The proposed method is implemented with careful engineering and integrated into the LOAM algorithm, forming a complete and practical system ready to use. To benefit the community by serving a benchmark for loop closure, the entire system is made open source on Github [1].**

I. Introduction

With the capacity of estimating the 6 degrees of freedom (DOF) state, and meanwhile building the high precision maps of surrounding environments, SLAM methods using LiDAR sensors have been regarded as an accurate and reliable way for robotic perception. In the past years, LiDAR odometry and mapping (LOAM) have been successfully applied in the field of robotics, like self-driving car [1], autonomous drone [2, 3], field robots surveying and mapping [4, 5], etc. In this paper, we focus on the problem of developing a fast and complete loop closure system for laser-based SLAM systems.

Loop closure is an essential part of SLAM system, to estimate the long term accumulating drift caused by local feature matching. In a common paradigm of loop closure, the successful detection of loops plays an essential role. Loop detection is the ability of recognizing the previously visited places, by giving a measurement of similarity between any two places. For visual-slam methods, the loop closure considerably benefits from various largely available computer vision algorithms. For example, by utilizing the bag-of-words model [6, 7] and clustering the feature descriptors as words, the similarity between observations can be computed in the word space. This kind of method has been used in most of the state of the art visual SLAM system (e.g. [8, 9]) and have achieved great success in the past years.

Unlike the visual-SLAM, the relevant research of laser-based loop closure is rare, and it is surprisingly hard for us to find any available open soured solution which addresses this problem. We conclude these phenomenons as two main reasons: Firstly, compared to the camera sensors, the cost of LiDAR sensors are extremely expensive, preventing them in wider use. In most of the robotics navigation perception,

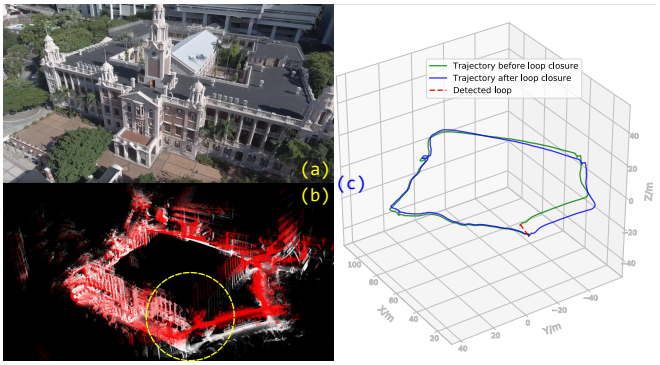


Fig. 1: An example of loop closure around the Main Building of the Hong Kong University (HKU). (a), the RGB image of the map area; (b), the red and withe points are off the map before and after loop closure, respectively; (c), the red dashed line indicates the detected loop, the green, and blue solid lines are the trajectories before and after loop closure, respectively.

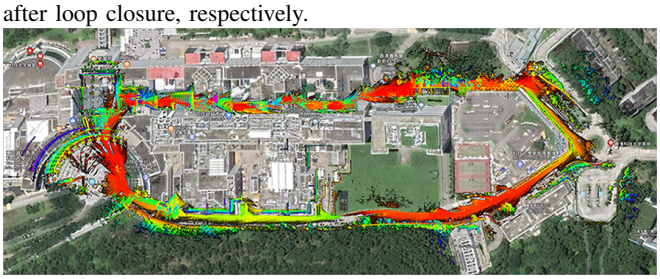


Fig. 2: The large scale loop closure of the Hong Kong University of Science and Technology (HKUST) campus. We align the point cloud map after loop closure with the satellite image. Our video is available at `https://youtu.be/fOSTJ_yLhFM` .

LiDARs is not always the first choice. Secondly, the problem of place recognition on point cloud is very challenging. Unlike 2D images containing rich information such as textures and colors, the available informations in point cloud are only the geometry shapes in 3D space.

In this paper, we develop a fast, complete loop closure system for LiDAR odometry and mapping (LOAM), consisting of fast loop detection, maps alignment, and pose graph optimization. We integrate the proposed loop closure method into a LOAM algorithm with Livox MID40[2] sensor, a high performance low cost LiDAR sensor easily available. Some of the results we obtain are shown in Fig. 1 and Fig. 2. To contribute to the development of laser-based slam methods, we will open source all the datasets and codes on Github[1].

II. Related work

Loop closure is widely found in visual-SLAM to correct its long-term drift. The commonly used pipeline mainly consists of three steps: First, local feature of a 2D images

J. Lin and F. Zhang are with the Department of Mechanical Engineering, Hong Kong University, Hong Kong SAR., China. {jiarong.lin, fuzhang}@hku.hk

[1] `https://github.com/hku-mars/loam_livox`

[2] `https://www.livoxtech.com/mid-40-and-mid-100`

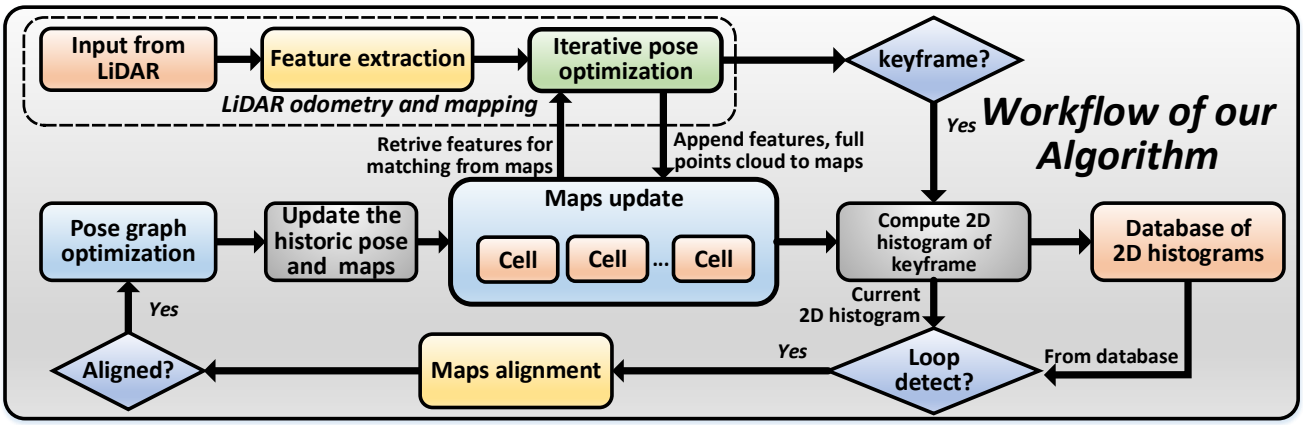


Fig. 3: The overview of our system.

is extracted by using handcrafted descriptors such as Scale-Invariant Feature Transforms (SIFT) [10], Binary Robust Independent Elementary Features (BRIEF) [11], Oriented Fast and Rotated BRIEF(ORB) [12], etc. Then, a bag-of-world model [6, 7] is used to cluster these features and build a dictionary to achieve loop detection. Finally, with the detected loop, a pose graph optimization is formulated and solved update the historic poses in the map.

Unlike the visual-SLAM, loop detection for point cloud data is still an open, challenging problem in laser-based SLAM. Bosse *et al* [13] achieve place recognition by directly extracting keypoints from the 3D point cloud and describe them with a handcrafted *3D Gestalt* descriptors. Then the keypoints vote for their nearest points and the scores are used to determine if a loop is detected. The similar method is also used in [14]. Magnusson *et al* [15] describe the appearance of 3D point cloud by utilizing the normal distribution transform (NDT), they compute the similarity of two scans from the histogram of the NDT descriptors.

Besides the hand-crafted descriptors, learning-based method has also been used in loop detection(or place recognitions) in recent years. For example, the SegMatch proposed by Dube *et al* [16] achieves place recognition by matching semantic features like buildings, tree, vehicles, etc. Angelina *et al* [17] realize place recognition by extracting the global descriptor from an end-to-end network, which is trained by combining the PointNet [18] and NetVLAD [19]. The learning-based method is usually computationally expensive, and the performances greatly rely on the dataset in the training process.

Although with these reviewed work, to our best knowledge, there is no open-sourced codes or dataset that benchmark the problem of loop closure for LOAM, which leaves some difficulties for readers on reproducing their works. To this purpose, we propose a complete loop closure system. The loop detection of our work is mainly inspired by the method of [15] and some of the adjustments are made in our scenarios. Due to the small FoV and special scanning pattern of our considered LiDAR sensor, we perform loop detection for an accumulated time of scans (i.e., the *keyframe*). To summarize, our contributions are threefold: (1) we develop a fast loop detection method to quickly measure the similarity of two keyframes; (2) we integrate our loop closure system, consisting of the loop detection, map alignment, and pose optimization into an LiDAR odometry and mapping algorithm (LOAM) [20], leading to a complete and practical system ready to use; (3) we provide an available solution and paradigm for point cloud based loop closure by opening source our systems and datasets on Github.

III. System Overview

The workflow of our system is shown in Fig. 3, each new frame (i.e., scan) input from LiDAR is registered to the global map by LOAM algorithm [20]. If a specified number of frames have been received (e.g., 100 frames), a keyframe created, which forms a small local map patch. The raw points, which were registered to the cells of the global map (Section IV) by LOAM, corresponding to the new keyframe are retrieved to compute its 2D histogram (see Section V). The computed 2D histogram is compared to the database, which contains 2D histograms of the global map consisting of all the past keyframes, to detect a possible loop (see Section VI). Meanwhile, the new keyframe 2D histogram is added to the database for the use of next keyframe. Once a loop is detected, the keyframe is aligned with global map and a pose graph optimization is performed to correct the drift in the global map.

IV. Map and Cell

In this section, we will introduce two key elements of our algorithm, the maps and cell. For conveniently, We use $\mathcal{M}$ and $\mathcal{C}$ denote the map and cell, respectively.

A. Cell

A cell is a small cube of a fixed size (i.e., S_x, S_y and S_z in x, y and z directions) by partitioning the 3D space. It is represented by its center location $\mathcal{C}_c$ and created by the first point $\mathbf{P}_i = [\mathbf{P}_{i_x}, \mathbf{P}_{i_y}, \mathbf{P}_{i_z}]^T$ in it

$$\mathcal{C}_c = \begin{bmatrix} \text{round}(\mathbf{P}_{i_x}/S_x) * S_x + S_x/2 \\ \text{round}(\mathbf{P}_{i_y}/S_y) * S_y + S_y/2 \\ \text{round}(\mathbf{P}_{i_z}/S_z) * S_z + S_z/2 \end{bmatrix} \quad (1)$$

Let N denote the number of points located in a cell $\mathcal{C}_c$, the mean $\mathcal{C}_\mu$ and covariance $\mathcal{C}_\Sigma$ of this cell is:

$$\mathcal{C}_\mu = \frac{1}{N}\left(\sum_{i=1}^{N}\mathbf{P}_i\right) \tag{2}$$

$$\mathcal{C}_\Sigma = \frac{1}{N-1}\left(\sum_{i=1}^{N}(\mathbf{P}_i - \mathcal{C}_\mu)(\mathbf{P}_i - \mathcal{C}_\mu)^T\right) \tag{3}$$

Notice that the cell is a fixed partitioning of the 3D space are is constantly populated with new points. To speed up the computation of mean 2 and covariance 3, we derive its recursive form as follows. Denote $\mathbf{P}_{N+1}$ the new point, N is the number of existing points in a cell with mean $\mathcal{C}'_\mu$ and covariance $\mathcal{C}'_\Sigma$. The mean $\mathcal{C}_\mu$ and covariance $\mathcal{C}_\Sigma$ of all the $N+1$ points in the cell are:

$$\mathcal{C}_\mu = \frac{1}{N+1}(N\mathcal{C}'_\mu + \mathbf{P}_{N+1}) \tag{4}$$

$$\begin{aligned}\mathcal{C}_\Sigma &= \frac{1}{N}\sum_{i=1}^{N+1}(\mathbf{P}_i - \mathcal{C}_\mu)(\mathbf{P}_i - \mathcal{C}_\mu)^T \\ &= \frac{1}{N}\sum_{i=1}^{N+1}(\mathbf{P}_i - \mathcal{C}'_\mu + \mathcal{C}'_\mu - \mathcal{C}_\mu)(\mathbf{P}_i - \mathcal{C}'_\mu + \mathcal{C}'_\mu - \mathcal{C}_\mu)^T \\ &= \frac{1}{N}\Big[(N-1)\mathcal{C}'_\Sigma + (\mathbf{P}_{N+1} - \mathcal{C}'_\mu)(\mathbf{P}_{N+1} - \mathcal{C}'_\mu)^T \\ &\quad + (N+1)(\mathcal{C}'_\mu - \mathcal{C}_\mu)(\mathcal{C}'_\mu - \mathcal{C}_\mu)^T \\ &\quad + 2\left(\mathcal{C}'_\mu - \mathcal{C}_\mu\right)\left(\mathbf{P}_{N+1} - \mathcal{C}'_\mu\right)^T\Big]\end{aligned} \tag{5}$$

Therefore, a cell $\mathcal{C}$ is composed of its static center $\mathcal{C}_c$, the dynamically updated mean $\mathcal{C}_\mu$ and covariance $\mathcal{C}_\Sigma$, and the raw points collection $\{\mathbf{P}_i\}$: $\mathcal{C} = (\mathcal{C}_c, \mathcal{C}_\mu, \mathcal{C}_\Sigma, \{\mathbf{P}_i\})$.

B. Map

The map $\boldsymbol{\mathcal{M}}$ is the collection of all raw points saved in cells. More specifically, $\boldsymbol{\mathcal{M}}$ consists of a hash table $\boldsymbol{\mathcal{H}}$ and a global octree $\boldsymbol{\mathcal{O}}$. The hash table $\boldsymbol{\mathcal{H}}$ enables to quickly find the specific cell according to its center $\mathcal{C}_c$. The octree $\boldsymbol{\mathcal{O}}$ enables to find out all cells located in the specific area of given range. These two are of significant importance in speeding up the maps alignments.

For any new added cell $\mathcal{C}$, we compute its hash index $\mathcal{H}(\mathcal{C}_c)$ using the XOR operation of hash index of its individual components: $(\mathcal{C}_{c_x}, \mathcal{C}_{c_y}, \mathcal{C}_{c_z})$. The computed hash index is then added to the hash table of the map $\boldsymbol{\mathcal{H}}$. Since the cell is a fixed partitioning of the 3D space, its center location $\mathcal{C}_c$ is static, requiring no update for existing entries in the hash table (the hash table is dynamically growing though).

The new added cell $\mathcal{C}$ is also added to the Octree $\boldsymbol{\mathcal{O}}$ according to its center location, similar to the OctoMap in [21]. Algorithm 1 illustrates the procedure of incrementally creating cells and maps from new frames.

Algorithm 1: Registration of new frame

Input : Points $\boldsymbol{\mathcal{P}}_k$ from k-th frame, Current map $\boldsymbol{\mathcal{M}}$, the pose $(\mathbf{R}_k, \mathbf{T}_k)$ estimated from LOAM algorithm

for *each* $\mathbf{P}_l \in \boldsymbol{\mathcal{P}}_k$ **do**
- Transform $\mathbf{P}_l$ to global frame by $\mathbf{P}_i = \mathbf{R}_k\mathbf{P}_l + \mathbf{T}_k$.
- Compute the cell center $\mathcal{C}_c$ from (1).
- Compute the hash index $\mathcal{H}(\mathcal{C}_c)$.
- **if** $\mathcal{H}(\mathcal{C}_c) \notin \boldsymbol{\mathcal{H}}$ **then**
  - Create new cell $\mathcal{C}$ with center $\mathcal{C}_c$.
  - Insert $\mathcal{H}(\mathcal{C}_c)$ to hash table $\boldsymbol{\mathcal{H}}$ of map $\boldsymbol{\mathcal{M}}$.
  - Insert $\mathcal{C}_c$ to *Octotree* $\boldsymbol{\mathcal{O}}$ of map $\boldsymbol{\mathcal{M}}$.
- Add $\mathbf{P}_i$ to $\mathcal{C}$.
- Update mean $\mathcal{C}_c$ of $\mathcal{C}$ using (4).
- Update covariance $\mathcal{C}_\Sigma$ of $\mathcal{C}$ using (5).

V. 2D HISTOGRAM OF ROTATION INVARIANCE

The main idea of our fast loop detection is that we use the 2D image-like histograms to roughly describe the keyframe. The 2D histogram describes the distribution of the Euler-angles of the feature direction in a keyframe.

A. The feature type and direction in a cell

As mentioned previously, each keyframe consists of a number of (e.g., 100) frames and each frame (i.e., scan) is partitioned into cells. For each cell, we determine the shape formed by its points and the associated feature direction (denoted as $\mathcal{C}_d$). Similar to [15], we perform eigenvalue decomposition on the covariance matrix $\mathcal{C}_\Sigma$ in (3):

$$\mathcal{C}_\Sigma\mathbf{V} = \mathbf{V}\boldsymbol{\Lambda} \tag{6}$$

where $\boldsymbol{\Lambda}$ is diagonal matrix with eigenvalues in descending order (i.e., $\lambda_1 \geq \lambda_2 \geq \lambda_3$). In practice, we only consider cells with 5 or more points to increase the robustness.

- ***Cell of plane shape***: If λ_2 is significantly larger than λ_3, we regard this cell as a plane shape and regard the plane normal as the feature direction, i.e., $\mathcal{C}_d = \mathbf{V}_3$ where $\mathbf{V}_3$ is the third column of the matrix $\mathbf{V}$.
- ***Cell of line shape***: If the cell is not a plane and λ_1 is significantly larger than λ_2, we regard this cell as a line shape and regard the line direction as the feature direction, i.e., $\mathcal{C}_d = \mathbf{V}_1$, the first column of $\mathbf{V}$.
- ***Cell with no feature***: A cell which is neither a line nor plane shape is not considered.

B. Rotation invariance

In order to make our feature descriptors invariant to arbitrary rotation of the keyframe, we rotate each feature direction $\mathcal{C}_d$ by multiplying it to an additional rotation matrix $\mathbf{R}$, and expect that most of the feature direction are lie on X-axis, and the secondary most are on Y-axis. Since plane feature is more reliable than line feature (e.g., the edge of plane feature are treated as a line feature), we use the feature

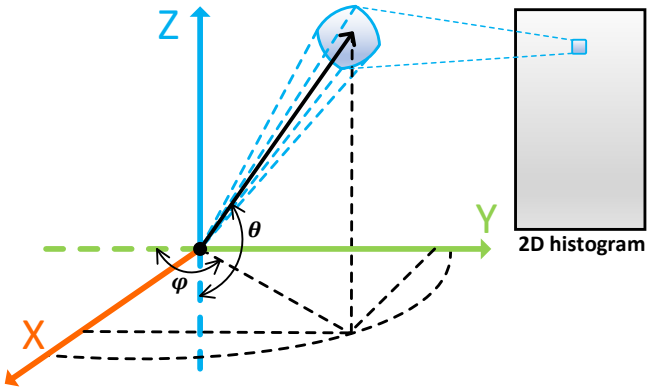


Fig. 4: The Euler angle of a feature direction and its contribution to the 2D histogram, each element of the 2D histogram is the number of feature directions with pitch θ and yaw ϕ located in the corresponding bin.

direction of plane cells to determine the rotation matrix $\mathbf{R}$. Similar to the previous sections, we compute the covariance $\mathbf{\Sigma}_d$ of all plane feature directions in a keyframe:

$$\mathbf{\Sigma}_d = \sum_{i=1}^{N} \boldsymbol{\mathcal{C}}_{i_d} \boldsymbol{\mathcal{C}}_{i_d}^T \tag{7}$$

where N is the number of plane cells, $\boldsymbol{\mathcal{C}}_{i_d}$ denotes the feature direction (i.e., plane normal) of the i-th plane cell. Similarly, the eigenvalue decomposition of $\mathbf{\Sigma}_d$ is:

$$\mathbf{\Sigma}_d \mathbf{V}_d = \mathbf{V}_d \mathbf{\Lambda}_d \tag{8}$$

where $\mathbf{\Lambda}_d$ is a diagonal matrix with eigenvalues in descending order ($\lambda_{d_1} \geq \lambda_{d_2} \geq \lambda_{d_3}$), $\mathbf{V}_d = \begin{bmatrix} \mathbf{V}_{d_1} & \mathbf{V}_{d_2} & \mathbf{V}_{d_3} \end{bmatrix}$ is the eigenvector matrix. Then, the rotation matrix $\mathbf{R}$ is determined as:

$$\mathbf{R} = \begin{bmatrix} \mathbf{V}_{d_1} & \mathbf{V}_{d_2} & \mathbf{V}_{d_1} \times \mathbf{V}_{d_2} \end{bmatrix}^T \tag{9}$$

After compute the rotation matrix $\mathbf{R}$, we apply the rotation transformation to all feature (both plane and line) directions.

Algorithm 2: Computing the 2D hist. of a keyframe

Input : Current keyframe $\boldsymbol{\mathcal{F}}$
Output: 2D Histogram $\mathbf{H_L}$ of line cell
2D Histogram $\mathbf{H_P}$ of plane cell
Start : $\mathbf{H_L} \leftarrow \mathbf{0}$, $\mathbf{H_P} \leftarrow \mathbf{0}$.
Compute rotation matrix $\mathbf{R}$ from Sect. V-B.
for *each* $\boldsymbol{\mathcal{C}} \in \boldsymbol{\mathcal{F}}$ **do**
  if $\boldsymbol{\mathcal{C}}$ *is a line shape* **then**
    $\boldsymbol{\mathcal{C}}_d \leftarrow \mathbf{R}\boldsymbol{\mathcal{C}}_d$
    Compute the pitch θ and yaw ϕ angle of $\boldsymbol{\mathcal{C}}_d$.
    $\mathbf{H_L}[\text{round}(\phi/3°), \text{round}(\theta/3°)]$ +=1
  if $\boldsymbol{\mathcal{C}}$ *is a plane shape* **then**
    $\boldsymbol{\mathcal{C}}_d \leftarrow \mathbf{R}\boldsymbol{\mathcal{C}}_d$
    Compute the pitch θ and yaw ϕ angle of $\boldsymbol{\mathcal{C}}_d$.
    $\mathbf{H_P}[\text{floor}(\phi/3°), \text{floor}(\theta/3°)]$ +=1
Gaussian blur $\mathbf{H_P}$ and $\mathbf{H_L}$.

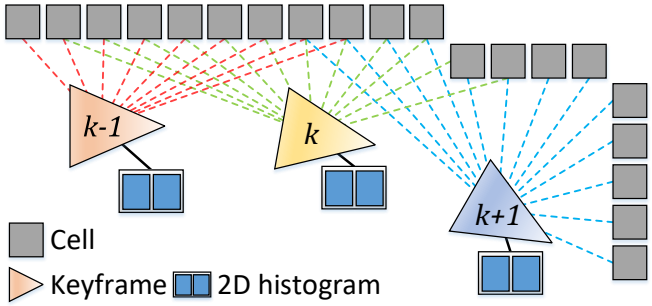


Fig. 5: A keyframe is consists of n (e.g., $n = 100$) frames (not shown), which then contains many cells. Each keyframe has two 2D histograms, one for line cells and the other one for plane cells.

C. 2D histogram of keyframe

With the rotation invariant feature directions of all cells in a keyframe, we compute the 2D histogram as follows:

Firstly, for a given feature direction $\boldsymbol{\mathcal{C}}_d = [\boldsymbol{\mathcal{C}}_{d_x}, \boldsymbol{\mathcal{C}}_{d_y}, \boldsymbol{\mathcal{C}}_{d_z}]$, we choose the direction with positive X components, i.e., $\boldsymbol{\mathcal{C}}_d = sign(\boldsymbol{\mathcal{C}}_{d_x}) \cdot \boldsymbol{\mathcal{C}}_d$. Then, the Euler angle of the feature direction is computed (see Fig. 4):

$$\theta = \sin^{-1}(\boldsymbol{\mathcal{C}}_{d_z}) + 90° \in [0°, 180°] \tag{10}$$

$$\phi = \tan^{-1}(\boldsymbol{\mathcal{C}}_{d_y} / \boldsymbol{\mathcal{C}}_{d_x}) + 90° \in [0°, 180°] \tag{11}$$

The 2D-histogram we use is a 60×60 matrix (have 3° resolution on both pitch and yaw angle), the elements of this matrix denote the number of line/plane cell with its pitch θ and yaw ϕ located in the corresponding bin. For example, i-th row, j-th column element, e_{ij}, is the number of cells with the angle of its feature direction satisfied:

$$j \times 3° \leq \theta < (j+1) \times 3°$$
$$i \times 3° \leq \phi < (i+1) \times 3°$$

To increase the robustness of the 2D histogram to possible noise, we apply a Gaussian blur on each 2D histogram we computed.

The complete algorithm of computing the 2D histogram with rotation invariance is shown in Algorithm. 2.

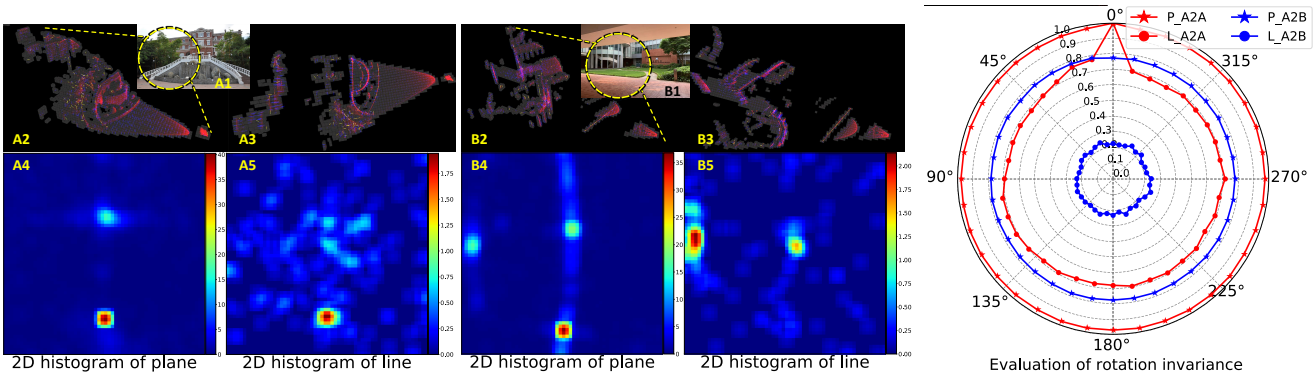
VI. Fast Loop Detection

A. Procedure of loop detection

As mentioned previously, we group n frames (e.g., $n = 100$) into a keyframe $\boldsymbol{\mathcal{F}}$. It can be viewed as a local patch of the global map $\boldsymbol{\mathcal{M}}$, and contains all of the cells appearing in the last n frames, as shown in Fig. 5. We compute the 2D histogram of a new keyframe $\boldsymbol{\mathcal{F}}$ and its similarity (Section VI. B) with all keyframes in the past to detect a loop. The keyframe with a detected loop is then matched to the map (Section VI. C) and the map is updated with a pose graph optimization (Section VI. D).

B. Similarity of two keyframes

For each newly added keyframe, we measure its similarity to all history keyframes. In this work, we use the normalized cross-correlation of 2D histograms to compute their similarity, which has been widely used in the field of computer vision (e.g., template matching, image tracking,



(a) The visualization of two keyframes (frame A and B), 2D histograms, and contained cells. Fig. A1 is the RGB image of the first scene. Fig. A2 and Fig. A3 are the side-view and bird view of the keyframe, respectively. In Fig. A2 and Fig. A3, the red points denote the raw point cloud in the keyframe, the white cubes denote the cells, the blue lines are the feature direction of plane cells, and the yellow lines are the feature direction of line cells. Fig. A4 and Fig. A5 are the 2D histogram (pixels are colored by their values) of plane and line features, respectively. The arrangement of Fig. B1∼B5 is the same as Fig. A1∼A5.

(b) The similarity in plane features between frame A and A ("P_A2A"), and between frame A and B ("P_A2B"), and in line features between frame A and A ("L_A2A"), and between frame A and B ("L_A2B"). Polar distance is the similarity level while ploar angle is the magnitude of random rotations.

Fig. 6: The visualization of keyframe, cells, and 2D histograms (a) and the evaluation of rotation invariance (b).

etc.). The similarity $S(\mathbf{H}_1, \mathbf{H}_2)$ of two 2D histogram $\mathbf{H}_1, \mathbf{H}_2$ is computed as:

$$S(\mathbf{H}_1, \mathbf{H}_2) = \frac{\sum_I (\mathbf{H}_1(I) - \bar{\mathbf{H}}_1)(\mathbf{H}_2(I) - \bar{\mathbf{H}}_2)}{\sqrt{\sum_I \left(\mathbf{H}_1(I) - \bar{\mathbf{H}}_1\right)^2 \sum_I \left(\mathbf{H}_2(I) - \bar{\mathbf{H}}_2\right)^2}} \tag{12}$$

where $\bar{\mathbf{H}}_k = \frac{1}{N}\sum_I \mathbf{H}_k(I)$ is the mean of $\mathbf{H}_k$ and $I = (i, j)$ is the index of the element in $\mathbf{H}_k$. If the similarity $S(\mathbf{H}_1, \mathbf{H}_2)$ between two keyframes is higher than threshold (e.g., 0.90 for plane and 0.65 for line), a loop is thought to be detected.

C. Maps alignment

After the successful detecting of a loop, we perform maps alignment to compute the relative pose between two keyframes. The problem of maps alignment can be viewed as the registration between the target point cloud and source point cloud, as their work of [22].

Since we have classified the cell of linear shape and planar shape in our LOAM algorithm [20], we use the feature of edge-to-edge and planar-to-planar to iteratively solve the relative pose.

After the alignment, if the average distance of the points of edge/plane feature on is close enough to the edge/plane feature (distance less than $0.1m$), we regard these two maps are aligned.

D. Pose graph optimization

As the workflow is shown in Fig. 3, once the two keyframes are aligned, we perform the pose graph optimization following the method in [23, 24]. We implement the graph optimization using the Google *ceres-solver*[3]. After optimizing the pose graph, we update all the cells in the entire map by recomputing the contained points, the points' mean and covariance.

[3]http://ceres-solver.org/

VII. Results

A. Visualization of keyframe, cells, and 2D histograms

We visualize the two keyframes, their associated 2D histograms local maps, and contained cells in Fig. 6(a). This figure shows that the 2D histogram of the two different scenes are very distinctive.

B. Rotation invariance

We evaluate the rotation invariance of our loop detection method by computing the similarity of the two scenes in Fig. 6(a) with random rotations. For each level of rotation, we generate 50 random rotation matrix of random directions but the same magnitude, rotate one of the two scenes by the generated rotation matrix, and compute the average similarity among all the 50 rotations of the same magnitude. The similarity of keyframe A to itself, keyframe B to itself, and keyframe A to keyframe B are shown as Fig. 6(b). It can be seen that, the similarity of plane features almost hold the same under different or rotation magnitude, and the similarity of the same keyframe (with arbitrary rotation) is constantly higher than the similarity of different keyframes. For line features, although the similarity of the same keyframe slightly drops when rotation takes place, it is still significantly higher than the similarity of different keyframes.

C. Time of computation

We evaluate the time consumption or each step of our system on two platforms: the desktop PC (with *i7-9700K*) and onboard-computer (DJI manifold2[4] with *i7-8550U*). The averge running time of our algorithm run on *HKUST large scale dataset* (the first column of Fig. 7) are shown in Table. I, where we can see our proposed method is fast and suitable for real time scenenarios on both platforms.

[4]https://www.dji.com/cn/manifold-2

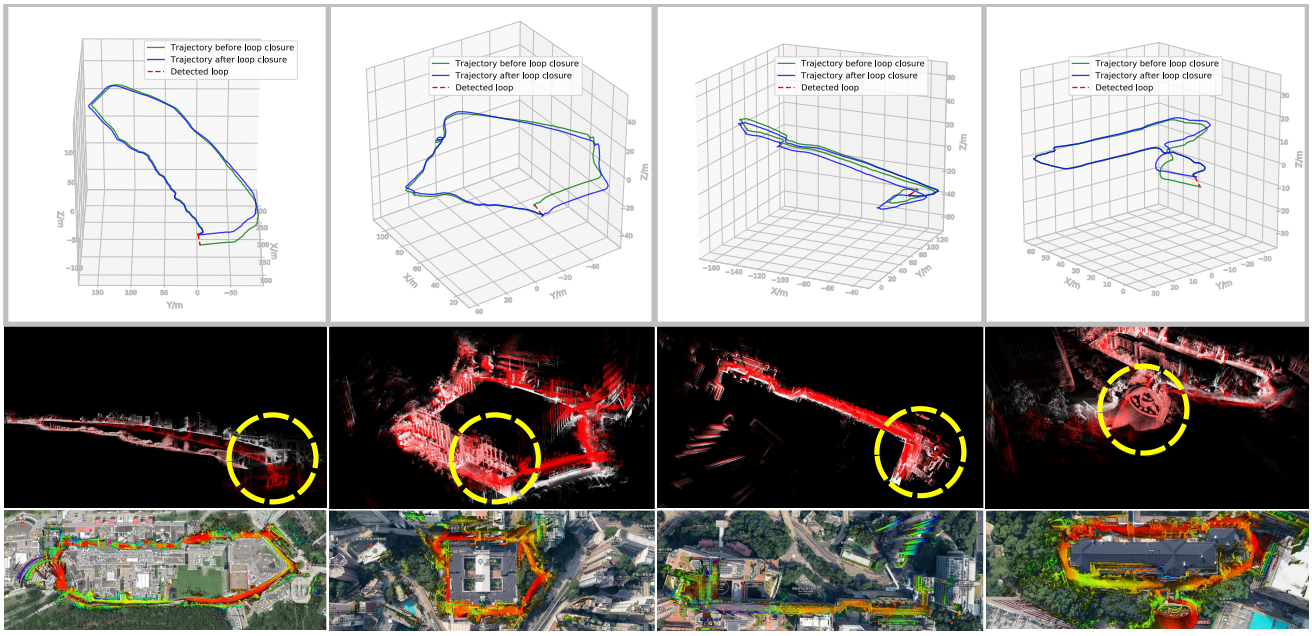


Fig. 7: We test our algorithm on four datasets, which are all sampled by Livox-MID40. The first one is a large scale dataset sampled in HKUST campus; The second one is sampled around a square building (main building of HKU); The third one is sampled indoor consisting of two long corridors in two neighboring floors. The fourth one is sampled around a rectangular building (Chong Yuet Ming Cultural Center in HKU) with many natural objects, such as trees, stairs, sculptures, etc.

	2D histogram computing	Maps alignment	Similarity of two maps
Desktop PC	1.18 ms	621 ms	13 μs
Onboard-computer	1.48 ms	931 ms	16 μs

TABLE I: The time table or our system run on two platforms.

D. Large scale loop closure results

We test our algorithm on four datasets in Fig. 7, where the first row is the comparison of trajectory before (green solid line) and after (blue sold line) loop closure, the red dashed lines indicate the detected loop. The second row of figures is the comparison of the point cloud map before (red) and after loop closure (white), where we can see the loop closure can effectively reduce the drift of LiDAR odometry and mapping (especially in the areas inside yellow circle). We align our point cloud after loop closure with Goolge maps in the third row, where we can see the alignment is very accurate, showing that the accuracy of our system is of high precision.

VIII. Conclusion

This paper presented a fast, complete, point cloud based loop closure for LiDAR odometry and mapping, we develop a loop detection method which can quickly evaluate the similarity of two keyframes from the 2D histogram of plane and line features in the keyframe. We test our system on four datasets and the results demonstrate that our loop closure can significantly reduce the long-term drift error. We open sourced all of our datasets and codes on Github to serve as an available solution and paradigm for point cloud based loop closure research in the future.

References

[1] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt, *et al.*, "Towards fully autonomous driving: Systems and algorithms," in *2011 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2011, pp. 163–168.

[2] A. Bry, A. Bachrach, and N. Roy, "State estimation for aggressive flight in gps-denied environments using onboard sensing," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 1–8.

[3] F. Gao, W. Wu, W. Gao, and S. Shen, "Flying on point clouds: Online trajectory generation and autonomous navigation for quadrotors in cluttered environments," *Journal of Field Robotics*, vol. 36, no. 4, pp. 710–733, 2019.

[4] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann, "6d slam3d mapping outdoor environments," *Journal of Field Robotics*, vol. 24, no. 8-9, pp. 699–722, 2007.

[5] B. Schwarz, "Lidar: Mapping the world in 3d," *Nature Photonics*, vol. 4, no. 7, p. 429, 2010.

[6] D. Gálvez-López and J. D. Tardos, "Bags of binary words for fast place recognition in image sequences," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, 2012.

[7] D. Filliat, "A visual bag of words method for interactive qualitative localization and mapping," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, 2007, pp. 3921–3926.

[8] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[9] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[10] D. G. Lowe *et al.*, "Object recognition from local scale-invariant features." in *iccv*, vol. 99, no. 2, 1999, pp. 1150–1157.

[11] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," in *European conference on computer vision*. Springer, 2010, pp. 778–792.

[12] E. Rublee, V. Rabaud, K. Konolige, and G. R. Bradski, "Orb: An efficient alternative to sift or surf." in *ICCV*, vol. 11, no. 1. Citeseer, 2011, p. 2.

[13] M. Bosse and R. Zlot, "Place recognition using keypoint voting in large 3d lidar datasets," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 2677–2684.

[14] A. Gawel, T. Cieslewski, R. Dubé, M. Bosse, R. Siegwart, and J. Nieto, "Structure-based vision-laser matching," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 182–188.

[15] M. Magnusson, H. Andreasson, A. Nuchter, and A. J. Lilienthal, "Appearance-based loop detection from 3d laser data using the normal distributions transform," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 23–28.

[16] R. Dubé, D. Dugas, E. Stumm, J. Nieto, R. Siegwart, and C. Cadena, "Segmatch: Segment based place recognition in 3d point clouds," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 5266–5272.

[17] M. Angelina Uy and G. Hee Lee, "Pointnetvlad: Deep point cloud based retrieval for large-scale place recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4470–4479.

[18] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 652–660.

[19] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "Netvlad: Cnn architecture for weakly supervised place recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 5297–5307.

[20] J. Lin and F. Zhang, "Loam_livox: A fast, robust, high-precision lidar odometry and mapping package for lidars of small fov," *arXiv preprint*, 2019.

[21] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous robots*, vol. 34, no. 3, pp. 189–206, 2013.

[22] K. Pulli, "Multiview registration for large data sets," in *Second International Conference on 3-D Digital Imaging and Modeling (Cat. No. PR00062)*. IEEE, 1999, pp. 160–168.

[23] E. Olson, J. Leonard, and S. Teller, "Fast iterative alignment of pose graphs with poor initial estimates," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE, 2006, pp. 2262–2269.

[24] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based slam," *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, pp. 31–43, 2010.