OPEN SOURCE AND AFFORDABLE TERRESTRIAL LASER SCANNER

Aung Bhone Pyae [1], Nyi Nyi Nyan Lin [1] *, Zwe Zaw Zaw [1]

[1] Myanmar Archaeology Association, Yangon, Myanmar - (aungbhonepyaemaeu, nyinyinyanlin.mm, zwezzaw)@gmail.com

Commission II

KEY WORDS: Archeology, Affordable, Open source, Terrestrial laser scanner, LiDAR, Point Cloud Library, Cultural heritage

ABSTRACT:

This paper explores the development of an affordable terrestrial laser scanner (TLS) system based on low-cost components and open source software. This paper enumerates components implemented and integrated in both hardware and software aspects. Primarily, the proposed system depends on Mid-40 LiDAR sensor and utilizes Point Cloud Library (PCL), Open Source Computer Vision Library (OpenCV) and Robotic Operating System (ROS) for system operation and data processing. Furthermore, this paper discusses sensor calibration and fusion methods available and implemented in the processing pipeline. The goal of this work is to produce an open source package of software system and hardware designs of a robust version of the TLS system that can be shared and easily reproduced by individuals and communities for their applications.

1. INTRODUCTION

Laser scanning is an essential technology in archeology and heritage preservation projects. Despite the extreme accuracy of survey-grade terrestrial laser scanners (TLS), most of them are expensive and could only be utilized in a well-funded situation. This limitation forbids archeologists and community-based activities from accessing this technology and its benefits. However, with recent progress in the Light Detection and Ranging (LiDAR) sensor industry, solid-state LiDAR sensors were being introduced to the market with significantly reduced prices and improved quality in the past few years. This paper explores the development of an affordable open source TLS system based on a low cost LiDAR sensor and open source software, and its technical feasibility.

For this work, a prototype TLS system has been developed, based on Livox Mid-40, Raspberry Pi computer, open source softwares and libraries such as Robotic Operating System (ROS), Point Cloud Library (PCL), Open Source Computer Vision Library (OpenCV), Point Data Abstraction Library (PDAL), CloudCompare and Eigen. In the following sections, the architecture of this prototype TLS system is explained and results from a site test scanning of a colonial-era heritage building in Yangon using this developed system is presented. Since this development work is an ongoing work, further experimentations on prospective approaches and modifications to the system pipeline and methodology presented in the current paper will be done in the future accordingly. From current results and progress, the technical feasibility of such a system can be tentatively considered possible despite the existence of some accuracy and performance setbacks.

Hardware designs, software packages and documentations developed in this work will be shared and updated at github repository[1].

2. RELATED WORKS

Implementations of Livox Mid-40 LiDAR sensor can be seen in mobile and Unmanned Aerial Vehicle (UAV) mapping systems. In (Lin and Zhang, 2020), a robust LiDAR Odometry and Mapping (LOAM) system using Livox LiDAR sensor is developed and demonstrated. Similar LOAM and Simultaneous Localization and Mapping (SLAM) systems (Wei et al., 2021) based on Livox LiDAR and additional sensors are demonstrated as well. However, LOAM/SLAM requires significant processing power due to the simultaneous feature extraction, registration and localization processes. In tight corners or feature poor environments, LOAM algorithms often fail to detect and register enough features failing to estimate and keep track of current sensor pose due to Mid-40's small field of view (FOV). On the other hand, LOAM systems implementing spindle-type 2D/3D LiDAR sensors do not suffer from such setbacks due to their 360° horizontal FOV coverage and repetitive scan line pattern.

Terrestrial laser scanner system developments similar to this work are (Bula, Derron and Mariethoz, 2020), (Zogg, 2008), (Wang et al., 2019), (Wang, Wang and Xie 2021) and (Eitel, Vierling and Magney, 2013). They utilize 2D 360° single/multi scan line LiDARs or Time of Flight (TOF) sensors with yaw/pitch mechanisms. This work adopts a similar approach and combines Mid-40 sensor with yaw and pitch drives to cover 360° spherical FOV.

3. DEVELOPED METHODOLOGY

The ideal affordable open source TLS system would contain components that are readily available in the consumer market, convenient to put together, affordable by individuals and community. Hardware and software components to build this proposed system are chosen with such mentioned criterias. They will ensure that the system is reproducible, modifiable and extendable. In this section, the development and technical concept of the developed TLS system will be presented.

In summary, Livox Mid-40 LiDAR sensor is interfaced with an onboard computer that hosts a set of subprograms in form of ROS nodes to process point cloud data with PCL. The LiDAR sensor is rotated in 3D space with stepper motor driven yawing and pitching mechanisms to provide a 360° spherical scan coverage. A camera is integrated as well to colorize the point cloud data output from LiDAR sensor. Calibration procedures are necessary to retrieve intrinsics, extrinsics and distortion parameters of the LiDAR sensor and camera. Additional supplementary sensors such as the Inertial Measurement Unit (IMU) and Real Time Kinematic/Global Navigation Satellite

[1] https://github.com/nyinyinyanlin/os-tls

Systems (RTK/GNSS) receivers are integrated to retrieve orientation and position of the TLS. The onboard computer hosts a web server that connects with the ROS process network to provide a human-machine interface to the operator over wireless network.

3.1 Hardware

As demonstrated in (figure 1), hardware components that made up this system are integrated.

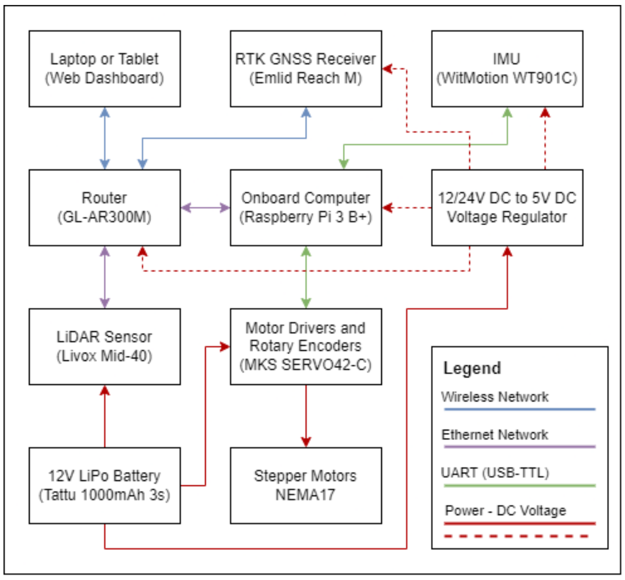

Laptop or Tablet (Web Dashboard)
RTK GNSS Receiver (Emlid Reach M)
IMU (WitMotion WT901C)
Router (GL-AR300M)
Onboard Computer (Raspberry Pi 3 B+)
12/24V DC to 5V DC Voltage Regulator
LiDAR Sensor (Livox Mid-40)
Motor Drivers and Rotary Encoders (MKS SERVO42-C)
12V LiPo Battery (Tattu 1000mAh 3s)
Stepper Motors NEMA17
Legend
Wireless Network
Ethernet Network
UART (USB-TTL)
Power - DC Voltage

Figure 1. Hardware integration diagram

3.1.1 Solid-state light detecting and ranging sensor

This work is primarily based on the cost effectiveness and high accuracy of Livox Mid-40 LiDAR sensor. It is a solid-state sensor with maximum range of 260 meters and a non-repetitive scanning pattern equivalent to 32-line products over 0.1 seconds integration time. The trade-off for a low-cost and high accuracy scanning is that the sensor projects a cone-shaped projection with a 38.4° circular FOV along the forward x-axis in 3D space. Yaw and pitch drives are essential to cover 360° FOV.

3.1.2 Onboard processing computer

A Raspberry Pi 3 B+ is implemented as the onboard computer due to its cheap price, GPIO sensor interfacing, built-in networking and moderate processing capacity. Alternative embedded computers with better computation performance and accelerated hardware such as Nvidia Jetson can be installed in the system instead. This computer performs data processing and sensor interfacing tasks as well as hosts a web server to provide web-based human-machine interface to the TLS.

3.1.3 Camera

A Raspberry Pi camera module v1.3 is installed above the LiDAR sensor on the same pitching frame for point cloud colorization and optionally for generating a panoramic image of the scanned environment. The camera contains a 5MP Omnivision 5647 CMOS sensor with a fixed focus configuration and 54° x 41° FOV. Camera is connected to the onboard computer through the Camera Serial Interface (CSI). Any alternative camera with similar FOV to Mid-40 sensor can be used as well.

3.1.4 Yawing and pitching mechanism

As the implemented LiDAR sensor projects only a cone-shaped FOV in 3-dimensional space, yawing and pitching mechanisms to rotate the sensor is necessary. In this system, two NEMA17 stepper motors are used to drive these yaw and pitch mechanisms through GT2 timing belts and pulleys systems with a reduction gear ratio of 82:30 as in (figure 2). Stepper motors are controlled with MKS-SERVO42C controllers which have built-in magnetic angular position encoders to serve as a feedback. The onboard computer communicates with motor drivers through the Universal Asynchronous Receiver-Transmitter (UART) bus. As the scanning process takes place, these mechanisms rotate the LiDAR sensor and camera in the 3-dimensional space.

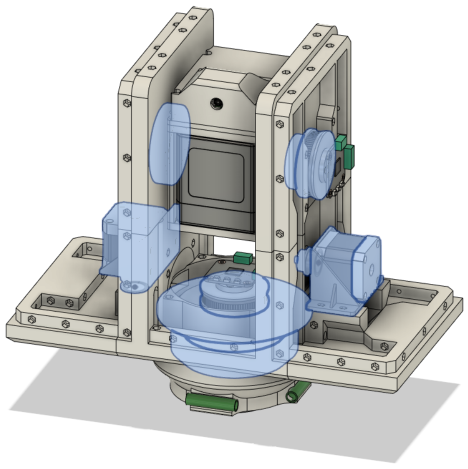


Figure 2. Stepper motors, gears and bearings of yawing and pitching mechanisms highlighted in blue

3.1.5 Inertial measurement unit

One IMU sensor is installed to detect the system’s absolute orientation in the 3-dimensional world coordinate system. A WitMotion WT901C 9-axis IMU which is based on MPU-9250 with built-in Kalman filtering and sensor fusion algorithms is installed on the yaw frame. The onboard computer retrieves orientation information from this IMU through the UART bus. This orientation information is useful in leveling the scanner and in compensating scanned point clouds’ orientation in the world coordinate system.

3.1.6 Real-time kinematic positioning and GNSS

Georeferencing of the scanned point cloud can be done in the post processing stage of the survey with position information obtained through indirect surveying method. However, an onboard RTK GNSS receiver is installed to obtain the precise position of the TLS system in the 3-dimensional world coordinate system. The Emlid Reach module receives Radio Technical Commission for Maritime Services (RTCM) correction messages and broadcasts position output over the wireless network. This position data can be integrated in real time transformation and translation of scanned point clouds or can be applied in the post processing stage. Position data from each module is retrieved over Transmission Control Protocol (TCP) and broadcasted in the ROS process network.

3.1.7 3D printed structure and enclosure

The structure and enclosure of the current TLS prototype (figure 3) is 3D printed and strengthened by embedding carbon fiber tubes. Due to the limitation of 3D print bed size, the structure is printed as multiple parts, fitted and secured together using insert screws and bolts. This approach introduces minor dimensional errors of 3D printed parts and inaccuracies in alignment which might affect the accuracy of transformation and alignment of consecutive point clouds.
Using precision Computer Numerical Control (CNC) milled metal parts instead of 3D printed ones will minimize these errors and improve the structural integrity of the TLS and the quality of data produced overall.

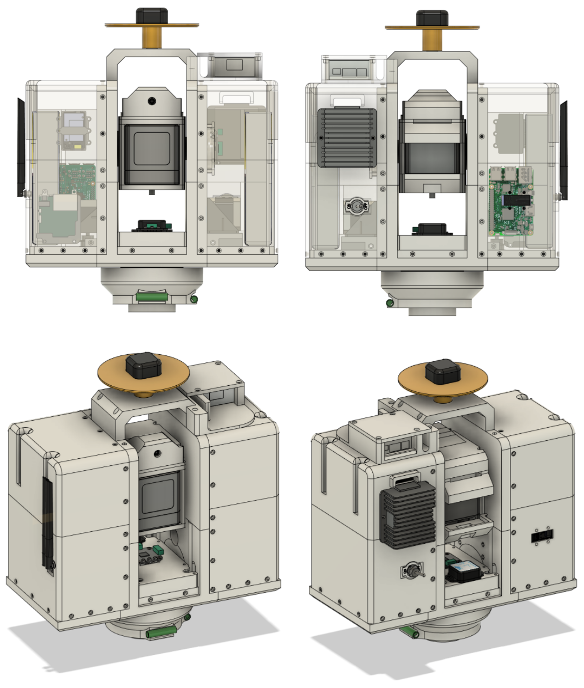


Figure 3. 3D CAD model of the developed TLS

3.1.8 Hardware miscellaneous

In order to facilitate the data transfer between the onboard computer, LiDAR sensor, RTK GNSS receiver and the human-machine web interface device, GL-AR300M mini router is installed onboard.
A DC-DC step down voltage regulator is also installed to convert DC 12V-24V input from LiPo battery to DC 5V voltage required by onboard computer and other components such as router, RTK GNSS receiver, etc. The LiDAR sensor and motor drivers are powered directly from the battery.

3.2 Software

ROS is the core software component of the developed TLS system. Sensor interfacing, point cloud data processing and motor control nodes are written in C++ and Python programming languages. They heavily utilize algorithm implementations in PCL and OpenCV for data processing. The Node.js server will act as a node in the ROS system and a web server to provide Application Programming Interface (API) endpoints and a web graphical user interface (GUI) for operating the TLS.

3.2.1 Operating system

Ubuntu MATE, a free and open source Linux and an official Ubuntu distribution, is installed as the operating system of the onboard computer. Although Raspbian operating system based on Debian is commonly used with Raspberry Pi computers, Ubuntu MATE 20.04.3 LTS (Focal Fossa) is installed instead because it supports Raspberry Pi 3 B+ with arm64 architecture variant which is also supported by ROS Noetic distribution.

3.2.2 Robotic operating system

ROS is implemented as the core component of the TLS as it facilitates the networking and data transfer between multiple processes and devices with well defined protocols and data structure. Additionally, multiple software components that perform sensor interfacing, data processing and visualization are already developed by the community and available as open source software packages in the ROS ecosystem. This reduces steps required to develop the system as well as increases the adaptability and expandability of the system and integration with robotic autonomous data acquisition systems in future development works.

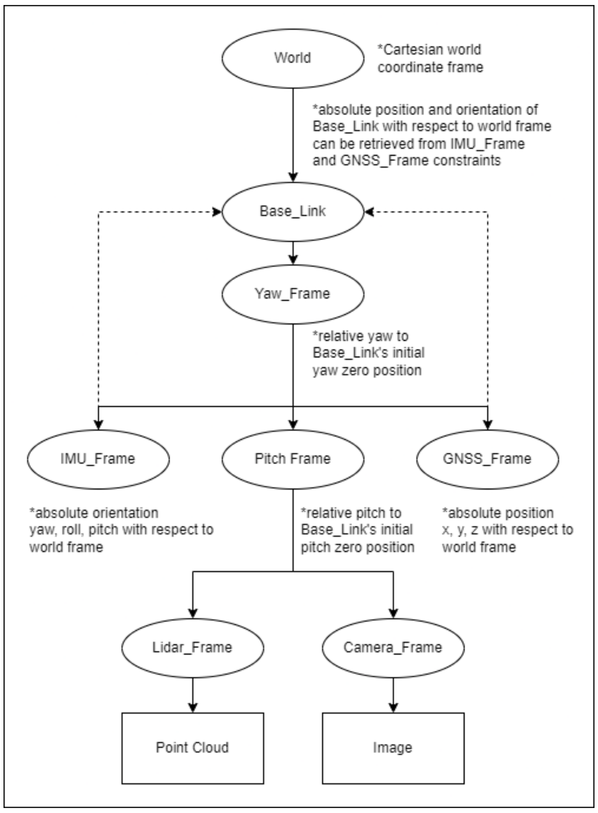

World
*Cartesian world coordinate frame
*absolute position and orientation of Base_Link with respect to world frame can be retrieved from IMU_Frame and GNSS_Frame constraints
Base_Link
Yaw_Frame
*relative yaw to Base_Link's initial yaw zero position
IMU_Frame
Pitch Frame
GNSS_Frame
*absolute orientation yaw, roll, pitch with respect to world frame
*relative pitch to Base_Link's initial pitch zero position
*absolute position x, y, z with respect to world frame
Lidar_Frame
Camera_Frame
Point Cloud
Image

Figure 4. TF2 coordinate frames with relation tree

3.2.3 TF2 library

TF2 (Foote 2013) is one of the major components of ROS that facilitates the automatic conversion and transformation of reference coordinate frames of different sensor poses and actuators with respect to one reference frames, transformation of sensor poses and sensed spatial information will have to be done frequently. TF2 keeps track of sensor poses and reference frames that exist in the system, and calculates transformations of reference frames with respect to each other. Reference frames and their relation tree is demonstrated in (figure 4).

3.2.4 Point cloud library and open source computer vision library

PCL is the primary component used to process point cloud data scanned by the LiDAR sensor. PCL is used to filter, transform and aggregate point clouds. PCL also provides implementations of feature extraction and alignment algorithms as well. OpenCV is used for image processing and point cloud colorization from the image stream of the camera.

3.2.5 Software miscellaneous

The onboard computer hosts a backend NodeJS server which serves a web dashboard to provide a human-machine interface to the TLS operator over wireless network. The server exposes API endpoints that in turn are connected with the ROS network through rosbridge_suite and rosnodejs. It also connects with a local SQLite3 database to keep record of settings and scanned projects. The human-machine interface is provided as a web dashboard accessible through a web browser. The dashboard depends on websockets, D3.js, roslibjs and ros3djs for visualization of point cloud data and communication with the ROS process network.
PM2 daemon process manager is used to monitor and manage various processes with different nature such as NodeJS server, ROS nodes and miscellaneous bash startup scripts altogether.

3.3 System pipeline

The real-time stream of point cloud data from the LiDAR sensor is continuously published in the ROS network by a sensor interfacing node. The filtering node performs a Statistical Outlier Removal (SOR) operation on this data to remove noises and rebroadcasts the cleaned point cloud to the system. The filtered point cloud is aggregated from sparse point clouds produced by Mid-40’s non-repetitive scanning pattern into a point cloud covering full FOV for single sensor pose within the set time duration between 3 to 10 seconds.
The aggregated point cloud is colorized by the colorizer node through fusing point cloud data and color pixel data from the image published by the camera interfacing node. The point cloud produced by Mid-40 has reflectance value of individual points attached but it has no color information. Point cloud colorization is done by projecting point cloud onto an undistorted image as a depth image and associating pixel by pixel color values. This color assigned image is reprojected back into 3D space as a point cloud. The intrinsic, extrinsic and distortion parameters of both camera and LiDAR sensor are obtained prior through calibration steps. From the sensor coordinate frame, this colorized point cloud is transformed into the coordinate frame of the TLS. This rigid transformation is calculated by TF2. The pose transformation matrix between sensors’ coordinate frames and the TLS coordinate frame is applied to the point cloud.
This transformed and colorized point cloud represents only a single FOV scan of current sensor pose. The yaw and pitch drives rotate the sensor pose incrementally with the same scan process repeated for full spherical scan coverage. Consecutive point clouds from each iteration are aggregated into a local scene point cloud by another aggregator node. Automatic point cloud alignment methods, which are not implemented in the current development stage, can be applied at this stage to produce a globally consistent and aligned point cloud. Data output at each stage of the pipeline can be visualized in the web user interface in real time and final point cloud data as Point Cloud Data (PCD) or LASer (LAS) file format can be downloaded through the web dashboard interface. The system pipeline can be summarized as presented in (figure 5).

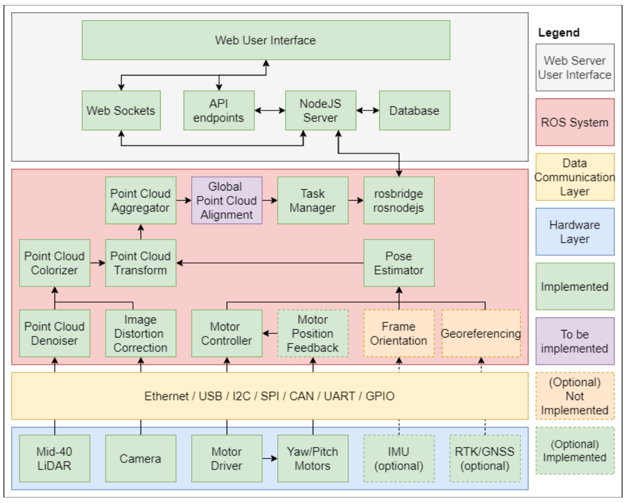


Figure 5. System pipeline overview diagram

3.4 LiDAR and camera calibration

Calibration of LiDAR and camera sensors installed is essential for sensor fusion and point cloud colorization. Camera intrinsic and distortion parameters are obtained through (Zhang, 2000) calibration method (figure 6) implemented as the camera calibration node (Bowman and Mihelich 2017).

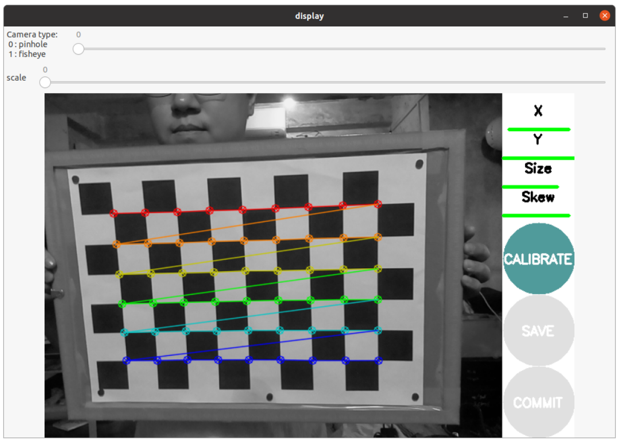
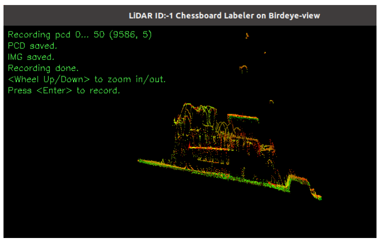


Figure 6. Camera calibration interface

Multiple calibration approaches to evaluate intrinsics and extrinsics of LiDAR sensor and camera for current hardware components exist. The calibration method (Robinson and SDK 2020) provided by the LiDAR sensor vendor is tedious and produces a suboptimal result. On the other hand, ACSC calibration method by (Cui et al., 2020) is fairly easy to perform (figure 7) and produces an acceptable result as demonstrated in (figure 12). However, it requires a calibration marker (figure 8) based on the checkerboard pattern and machine learning software package dependencies to work. Automatic calibration approach provided by (Yuan et al. 2021) produces the optimal result with a straightforward procedure and without the need for a target marker. Extrinsic parameters of both sensors are obtained from ROS implementation of (Yuan et al. 2021) calibration method.



```
Recording pcd 0... 50 (9586, 5)
PCD saved.
IMG saved.
Recording done.
<Wheel Up/Down> to zoom in/out.
Press <Enter> to record.
```

Figure 7. ACSC calibration interface

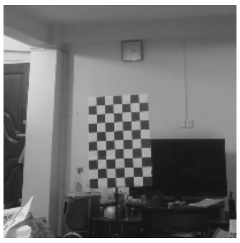


Figure 8. ACSC calibration target

3.5 Point cloud colorization

The point cloud data produced from the LiDAR sensor contains only the reflectance intensity value of each point in the cloud. Therefore, the point cloud is colorized using color information retrieved from the camera’s image stream. The relation between the coordinate frames of the LiDAR sensor and camera are retrieved as intrinsics and extrinsics parameters through calibration steps as discussed in the previous section. These calibration parameters as a transformation matrix and camera distortion parameters are applied in OpenCV that projects LiDAR scanned 3D points onto the 2D image plane of the camera as a depth image. Projected points as depth image pixels are assigned with color information from respective pixels of the camera image. These projected points with the color information are associated back to the point cloud in 3D space respectively again. This colorized point cloud is broadcasted in the ROS process network for further processing and transformation.

3.6 Automatic point cloud alignment

Although point clouds from consecutive poses are rotated and optimally aligned initially, drifts and errors in alignments of the structure and body frame, yawing and pitching mechanisms can introduce offsets and drifts in position and rotation of point clouds. Minimizing these errors and improving the alignment accuracy can be achieved through automatic point cloud alignment methods such as iterative closest point cloud (ICP) as proposed in (Besl and McKay 1992) and its variants. Multiple approaches and implementations for this exist. For example, (Segal, Haehnel and Thrun 2009) for generalized ICP (GICP), (Koide et al. 2021) for voxelized GICP approach with GPU based implementation and (Pan et al. 2018) for global optimal matching and hybrid metric based approach.
Other approaches also exist, each with its own robustness, performance and time trade-offs with respect to processing computer hardware and the nature of the scanned environment. In addition to automatic point cloud alignment between consecutive poses, globally optimized and consistent alignment approaches such as (Liu and Zhang, 2021), (Gojcic et al. 2020) and (Theiler, Wegner and Schindler 2015) that takes all poses and point clouds in account and align them as a whole with constraints can be implemented to produce an accurate final point cloud. However, in the current state of the development, fine alignment using CloudCompare ICP tool is manually applied in the post processing stage.

3.7 Georeferencing

Although the direct georeferencing and real time transformation of scanned point clouds with position and orientation information received from RTK GNSS and IMU is possible, it would require a precise Inertial Navigation System (INS)/IMU with extra calibration steps such as boresight and/or lever arm calibrations. At the current stage of development, indirect georeferencing through control points or lines as (dos Santos, Dal Poz and Khoshelham, 2013) demonstrated is proposed to be adopted and georeferenced in the post-processing stage. Both CloudCompare and PDAL can be used to georeference the point cloud either manually or programmatically.

3.8 Test measurement

To test the robustness of the prototype TLS, a test scan (figure 9) of the Myanma Port Authority building, a Yangon City Heritage List designated colonial-era heritage building built in 1920, is carried out. Facades of this building are scanned with a 210° horizontal and 150° vertical FOV coverage at three locations marked by red dots in (figure 10). Rotary encoders were not yet installed in yawing and pitching mechanisms at the time of test measurement and therefore, no angular position feedback is applied in compensating for point cloud transformation. For each consecutive sensor pose with 15° FOV overlap, the point data stream is captured for 3 seconds, accumulating about 300,000 raw points. It takes approximately 15 seconds per pose and 17 minutes in total per TLS location capturing and processing about 65 consecutive point clouds. These point clouds are merged online. Resulting three point clouds are coarse aligned manually and fine aligned with the ICP tool in CloudCompare.

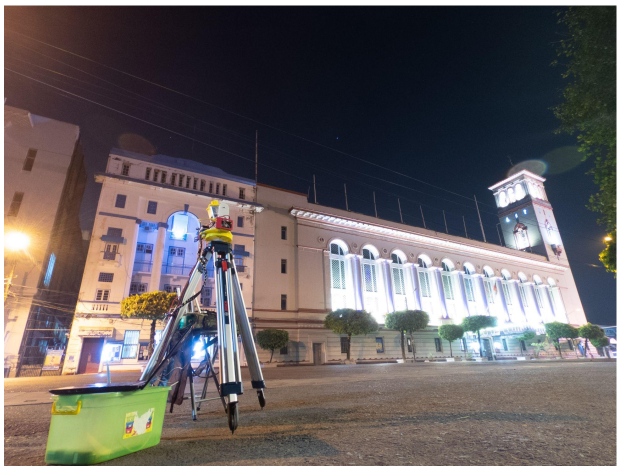


Figure 9. Test setup at Myanma Port Authority building

4. RESULTS

4.1 Site test result

While offline alignment of three point clouds from three TLS locations (figure 10 and 11) are satisfactory overall, minor misalignments between consecutive sensor pose scans of each TLS location exist mainly due to drifts and errors produced from yaw and pitch mechanisms as no angular position feedback is implemented at this stage. Final filtered and merged point cloud contains about 17.5 million points. Colorized point clouds based on the ACSC method are discarded due to the colorization error which is demonstrated in the following section.

4.2 Point cloud colorization result

As demonstrated in (figure 12) below, the colorized point cloud from LiDAR and camera using the ACSC calibration method displays an noticeable error. In comparison, (Yuan et al. 2021) calibration method produced a better colorization result than the (Cui et al., 2020) method. The reflectance intensity and colorized point clouds with (Yuan et al. 2021) calibration method is presented in (figure 13).

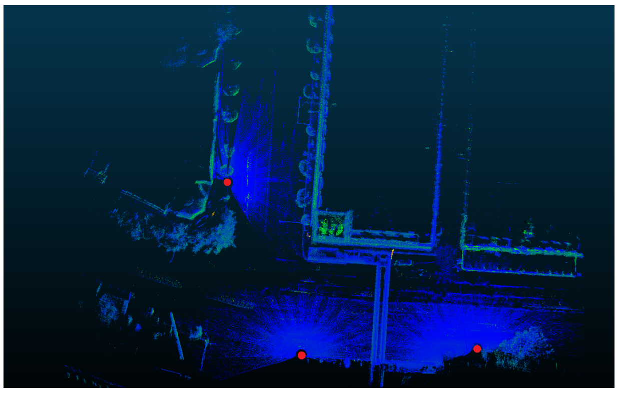


Figure 10. Top view of test scan point cloud

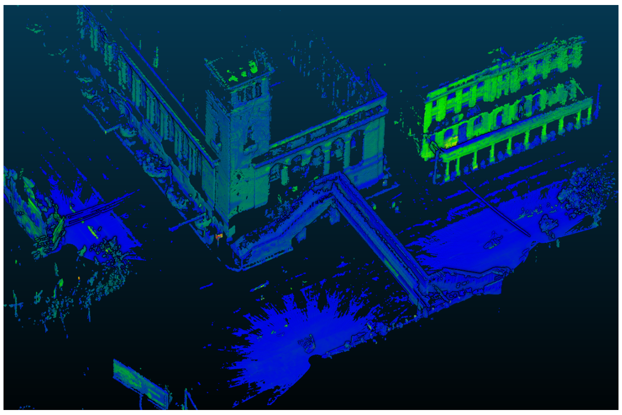


Figure 11. Perspective view of test scan point cloud

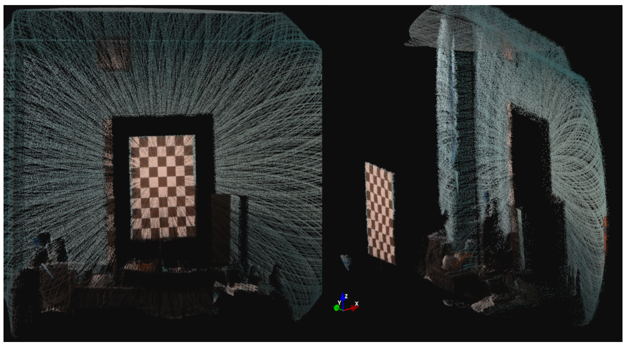


Figure 12. Point cloud colorized using (Cui et al., 2020)

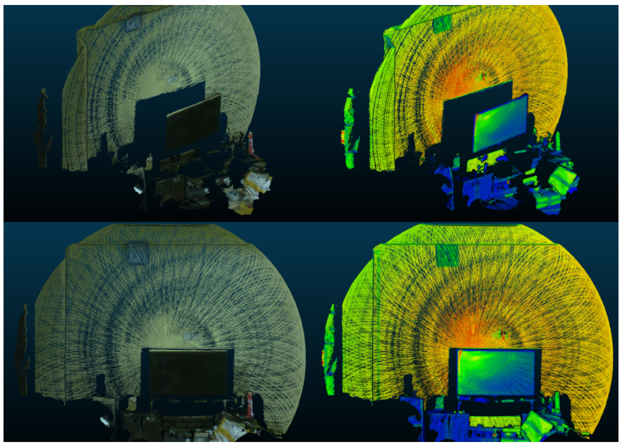


Figure 13. Point cloud colorized using (Yuan et al. 2021)

4.3 Cost breakdown analysis and comparison

(Table 1) shows the cost breakdown of the developed TLS system with minimal performance and a cost comparison with the system demonstrated in (Bula, Derron and Mariethoz, 2020) due to common focus on the low-cost TLS system.

Components & Approx. Cost	**Final TLS System**	**(Bula, Derron and Mariethoz, 2020)**
LiDAR	600 (Mid-40)	4,000 (VLP-16)
Yaw/Pitch Mechanism	80 (NEMA17 Stepper, MKS Servo42-C)	400 (Genie Mini)
Camera	30 (Raspberry Pi Camera Module V1.3)	-
Onboard Computer	50 (Raspberry Pi 3 B+)	-
GNSS/RTK	300 (Reach Module)	-
IMU	40 (WitMotion WT901C)	-
Power	100 (Tattu 3s 10000mAh, Voltage Regulator)	-
Tripod, Frame & Misc.	550	300
Total (USD)	1,750	4,700

Table 1. Cost breakdown and comparison

4.4 Software components and licenses

Essential software components that make up this system are adopted based on their free and open-sourceness as this development work focuses on being open source and accessible. (Table 2) enumerates the core software components and libraries that are used by this system to be functional.

Component	**License Type**
Ubuntu Mate	Open Source GPL
ROS core, tf2, PDAL, camera_calibration, roslibjs, ros3djs	BSD
Boost	BSL-1.0
OpenCV, Thread Building Blocks	Apache 2.0 and 3-clause BSD
CloudCompare, v4l-utils	GNU LGPL v2.0 and GNU GPL v2.0
Eigen	MPL2 and LGPL3+
PCL, PCD (file format), Ceres solver, rosbridge_suite, reach_rs_ros_driver, scipy, scikit-learn, pyserial	3-clause BSD
Livox ROS Driver, Bootstrap	MIT, Apache 2.0, BSL-1.0
lidar_camera_calib, ACSC	GNU GPL v2.0
PM2	AGPL-3.0
Node.js, Express, jQuery	MIT
SQLite	Public Domain
D3.js	ISC
LAS (file format)	No license required

Table 2. Software components and licenses

5. CONCLUSIONS AND FUTURE WORKS

Our preliminary results from current development progress showed that the development of an open source TLS system based on a low cost solid-state LiDAR sensor, consumer market components and open source software is technically feasible with the right combination of system components and processing pipeline, and applicable for non-critical 3D laser scanning documentation projects with acceptable data quality.
Experimentation with prospective automatic point cloud alignment methods will be carried out. The most robust approach will be implemented with respect to the computation performance and hardware of the onboard computer.
Direct georeferencing approaches by integrating position and orientation information from IMU and RTK GNSS in real time will be explored, experimented with and a robust approach will be integrated in the system pipeline. Further modifications and improvements on yawing and pitching mechanisms will also have to be done in order to improve the rotational accuracy and integrity of mechanisms.
Current work done lacks measurement analysis of the TLS and performance analysis of data processing implementations in the system pipeline. Precise calibration and measurement analysis of the TLS and performance analysis will be performed in a systematically setup controlled environment. Analysis results will be reported in future along with software/hardware updates.

6. ACKNOWLEDGEMENTS

This development work is done by authors as Yangon Binärhaus team in affiliation with Myanmar Archeology Association. This is initiated as an Innovative Idea award winning idea from the TechCul Ideathon organized by UNESCO Bangkok and FOSSASIA in January 2021. We thank and acknowledge respective persons from Myanmar Archeology Association, Ubuntu Myanmar LoCo team, TechCul, UNESCO Bangkok and FOSSASIA for their mentorship and support which encouraged our work.

7. CONFLICT OF INTEREST STATEMENT

We hereby declare that we have no affiliations with any organization or entity in financial or non-financial interests with an exception to UNESCO TechCul. We confirm and report that the affiliation with the said organization pertains to the financial or non-financial interest in the subject matter of the materials discussed in this paper; we received prize money from UNESCO TechCul as a gesture of support to realize the idea into an open source solution in the cultural and heritage sector.

REFERENCES

Besl, P.J., and Neil D. McKay. 1992. A Method For Registration Of 3-D Shapes. *IEEE Transactions On Pattern Analysis And Machine Intelligence* 14 (2): 239-256. doi:10.1109/34.121791.

Bowman, James, and Patrick Mihelich. 2017. Camera_Calibration. *Github*. https://github.com/ros-perception/image_pipeline/tree/noetic/camera_calibration.

Bula, Jason, Marc-Henri Derron, and Gregoire Mariethoz, 2020. Dense Point Cloud Acquisition With A Low-Cost Velodyne VLP-16. *Geoscientific Instrumentation, Methods And Data Systems* 9 (2): 385-396. doi:10.5194/gi-9-385-2020.

Cui, Jiahe, Jianwei Niu, Zhenchao Ouyang, Yun Qin He and Dian Liu. 2020. ACSC: Automatic Calibration For Non-Repetitive Scanning Solid-State LiDAR And Camera Systems. *ArXiv* abs/2011.08516v1

dos Santos, Daniel R., Aluir P. Dal Poz, and Kourosh Khoshelham, 2013. Indirect Georeferencing Of Terrestrial Laser Scanning Data Using Control Lines. *The Photogrammetric Record* 28 (143): 276-292. doi:10.1111/phor.12027.

Eitel, Jan U.H., Lee A. Vierling, and Troy S. Magney, 2013. A Lightweight, Low Cost Autonomously Operating Terrestrial Laser Scanner For Quantifying And Monitoring Ecosystem Structural Dynamics. *Agricultural And Forest Meteorology* 180: 86-96. doi:10.1016/j.agrformet.2013.05.012.

Foote, Tully., 2013. Tf: The Transform Library. *2013 IEEE Conference On Technologies For Practical Robot Applications (Tepra)*. doi:10.1109/tepra.2013.6556373.

Gojcic, Zan, Caifa Zhou, Jan D. Wegner, Leonidas J. Guibas, and Tolga Birdal. 2020. End-to-End Globally Consistent Registration of Multiple Point Clouds. *ETH Zürich*. doi:10.3929/ETHZ-B-000458888.

Koide, Kenji, Masashi Yokozuka, Shuji Oishi, and Atsuhiko Banno, 2021. Voxelized GICP For Fast And Accurate 3D Point Cloud Registration. *2021 IEEE International Conference On Robotics And Automation (ICRA)*. doi:10.1109/icra48506.2021.9560835.

Lin, Jiarong, and Fu Zhang, 2020. Loam Livox: A Fast, Robust, High-Precision Lidar Odometry And Mapping Package For Lidars Of Small Fov. *2020 IEEE International Conference On Robotics And Automation (ICRA)*. doi:10.1109/icra40945.2020.9197440.

Liu, Zheng, and Fu Zhang, 2021. BALM: Bundle Adjustment For Lidar Mapping. *IEEE Robotics And Automation Letters* 6 (2): 3184-3191. doi:10.1109/lra.2021.3062815.

Pan, Yue, Bisheng Yang, Fuxun Liang, and Zhen Dong. 2018. Iterative Global Similarity Points: A Robust Coarse-To-Fine Integration Solution For Pairwise 3D Point Cloud Registration. *2018 International Conference On 3D Vision (3DV)*. doi:10.1109/3dv.2018.00030.

Robinson, Brian, and Livox SDK. 2020. Calibrate The Extrinsic Parameters Between Livox Lidar And Camera. *Github*. https://github.com/Livox-SDK/livox_camera_lidar_calibration.

Segal, A., D. Haehnel, and S. Thrun. 2009. Generalized-ICP. *Robotics: Science And Systems V*. doi:10.15607/rss.2009.v.021.

Theiler, Pascal Willy, Jan Dirk Wegner, and Konrad Schindler. 2015. Globally Consistent Registration Of Terrestrial Laser Scans Via Graph Optimization. *ISPRS Journal Of Photogrammetry And Remote Sensing* 109: 126-138. doi:10.1016/j.isprsjprs.2015.08.007.

Wang, Han, Chen Wang, and Lihua Xie. 2021. Lightweight 3-D Localization And Mapping For Solid-State Lidar. *IEEE Robotics And Automation Letters* 6 (2): 1801-1807. doi:10.1109/lra.2021.3060392.

Wang, Pei, Ronghao Li, Guochao Bu, and Rui Zhao, 2019. Automated Low-Cost Terrestrial Laser Scanner For Measuring Diameters At Breast Height And Heights Of Plantation Trees. *PLOS ONE* 14 (1): e0209888. doi:10.1371/journal.pone.0209888.

Wei, Weichen, Bijan Shirinzadeh, Rohan Nowell, Mohammadali Ghafarian, Mohamed M. A. Ammar, and Tianyao Shen. 2021. Enhancing Solid State Lidar Mapping With A 2D Spinning Lidar In Urban Scenario SLAM On Ground Vehicles. *Sensors* 21 (5): 1773. doi:10.3390/s21051773.

Yuan, Chongjian, Xiyuan Liu, Xiaoping Hong, and Fu Zhang, 2021. Pixel-Level Extrinsic Self Calibration Of High Resolution Lidar And Camera In Targetless Environments. *IEEE Robotics And Automation Letters* 6 (4): 7517-7524. doi:10.1109/lra.2021.3098923.

Zhang, Z., 2000. A Flexible New Technique For Camera Calibration. *IEEE Transactions On Pattern Analysis And Machine Intelligence* 22 (11): 1330-1334. doi:10.1109/34.888718.

Zogg, Hans-Martin. 2008. Investigations Of High Precision Terrestrial Laser Scanning With Emphasis On The Development Of A Robust Close-Range 3D-Laser Scanning System. *ETH Zürich*. doi:10.3929/ETHZ-A-005679006.