A Time- and Memory-Efficient Method for Axis Alignment of Very Large Point Clouds using Orthogonality in Buildings

Longyong WU*, Sou-Han CHEN, Meng SUN and Fan XUE

Department of Real Estate and Construction, The University of Hong Kong, Hong Kong (China SAR)

wulongyong@connect.hku.hk, hankchen@connect.hku.hk, sunmhku01@connect.hku.hk, xuef@hku.hk

Abstract -

In construction and operational stages of buildings and in computer vision and construction robotics, accurate alignment of LiDAR point clouds to a building's 3D coordinate axes is an essential preprocessing step. However, existing point cloud processing tools handle a point cloud in a single batch, which can lead to high memory usage or even failures for very large clouds. This study presents a chunk-based, orthogonality-enabled method for aligning very large building LiDAR point clouds to the building's coordinate in parallel computing. Our method leverages the random-access characteristic of LiDAR's LASer (*.las*) format to divide a cloud into multiple chunks, then processes each chunk, with the number of concurrently running tasks being limited. Subsequently, the global orientation is estimated by extracting the dominant directions of the building's interior and exterior orthogonal structures to rotate and align the input cloud. Experiments on a 22 GB real-world LiDAR point cloud demonstrated the effectiveness of our proposed method: It reduced peak memory consumption from exceeding 60.5 GB to 4.1 GB and achieved a 3.85-fold speedup over the sequential method. (The source code is shared at: `https://github.com/chunibyo-wly/laspy-parallel`)

Keywords -

Building and Construction; LiDAR; Point cloud processing; Orthogonal structure; Parallel computing

1 Introduction

Light Detection and Ranging (LiDAR) point clouds have become increasingly essential data sources in construction [1], surveying [2], computer vision [3], and robotics [4], thanks to LiDAR's high precision in measuring the 3D built environment. Yet, a LiDAR point cloud's local coordinate axes are usually inconsistent with the target building's or project's coordinate axes in 2D drawings, footprint on the map, or 3D building information modeling (BIM).

Axis alignment refers to aligning a point cloud to the project's local coordinate axes [5]. Axis alignment is an important step to enable many construction digitalization and automation applications, including existing condition modeling, 3D coordination, registration of 2D computer-aided design (CAD) drawings [6], scan-to-BIM [7], scan-vs-BIM for change detection [8], and construction progress monitoring [9]. In computer vision, many deep learning models for point cloud processing have been proposed and trained well on coordinate-aligned datasets [10], and some robots for building operation and maintenance also relied on axis alignment.

However, existing tools for aligning LiDAR cloud coordinates with buildings and construction often encounter challenges. Building floor areas and construction sites can span multiple thousand square meters, and modern LiDAR scanners produce 10,000–400,000 points per square meter; these factors result in LiDAR clouds that can reach up to billions of points and tens or even hundreds of gigabytes (GB). Existing 3D point cloud processing tools, such as `Open3D` [11] and `pypcd4`, always load the full cloud data into computer memory in a single batch. This one-batch processing of very large LiDAR clouds is memory-intensive and sometimes exceeds the capacity of computers, which are equipped with no more than 32 GB Random-access memory (RAM) in most cases. When memory demand exceeds system capacity, the operating system may terminate the process, causing the computation to fail and limiting the scalability of LiDAR data analysis.

To mitigate the problem of exceeding system memory capacity in axis alignment, this study presents a chunk-based, orthogonality-enabled parallel computing method. The method aims to achieve relatively efficient axis alignment for very large point clouds. The method leverages the random-access characteristic of LiDAR's LASer (*.las*) format to divide an input LiDAR point cloud into partitioned chunks, then processes them in parallel with controlled concurrency. To ensure balanced orthogonality detection, the point distances in each chunk are normalized (down-sampled) to 2 cm, and point normals are estimated from the normalized points. Finally, the method automatically estimates and optimizes the dominant orthogonal structural directions of LiDAR point clouds, in order to rotate the point cloud and align the coordinates. Overall, this parallel computing utilizing orthogonality in buildings can achieve better time and memory efficiency on daily per-

sonal computers.

The contribution of this paper is twofold:

1. A novel chunk-based, orthogonality-enabled parallel computing method for axis alignment of very large LiDAR point clouds, and

2. Preliminary evidence on the boosted time and memory efficiencies of the present method from experiments on real-world very large data.

2 Related Works

LiDAR has been widely applied in the construction industry [12, 13] for various applications, ranging from semantic enrichment of BIM models to derivative-free optimization-based (DFO) 3D reconstruction of architectural structures. For instance, Xue et al. [14] is the first to transform architectural symmetry detection from LiDAR point clouds as a nonlinear optimization problem and proposed a DFO approach to solve it. Furthermore, Xue et al. [15] extended the DFO method to register a small-scale indoor LiDAR point clouds with BIM components, enabling the reconstruction of detailed indoor scenes.

Although existing studies have demonstrated LiDAR point clouds' applications in 3D modeling and semantic enrichment, most have been limited to small-scale or exterior-level point clouds [14, 16], due to computational and memory limits. In practice, LiDAR point clouds collected from construction sites often contain millions of interior points, posing significant challenges for preprocessing using existing methods. Nevertheless, these interior point clouds are valuable for analyzing architectural symmetries and skeletons.

Orthogonality, a common characteristic of real-world architectural forms and designs, offers researchers an opportunity to understand and reconstruct urban point clouds [17]. For instance, Wu et al. [12] applied orthogonality and symmetry constraints to efficiently reconstruct 3D models from facade point clouds, and later extended this framework to support morphological analysis. This inherent architectural characteristic enables the development of chunk-based, orthogonality-enabled methods for the axis alignment of very large LiDAR point clouds.

In summary, it is of interest to design a chunk-based, orthogonality-enabled method for axis alignment of very large LiDAR point clouds, for LiDAR and construction digitalization practitioners who handle very large indoor or outdoor scenes in a time- and memory-efficient manner.

3 Our Method for Processing Very Large Point Clouds

3.1 LiDAR Scanning Principles and the LASer Format

LiDAR has been widely used in construction, robotics and mapping. It actively emits laser pulses and measures the time delay of their reflections from surfaces to estimate distances. Compared with passive sensors such as cameras, LiDAR is less affected by lighting conditions and can provide depth information even in poor lighting. After obtaining raw distance measurements, LiDAR point clouds can be combined with simultaneous localization and mapping algorithms to automatically estimate the scanner's position and orientation in unknown environments, reconstructing 3D environment with rich textures and high-precision geometric details [18]. This capability makes LiDAR a powerful tool for change detection, building modeling, and progress monitoring [13].

To efficiently manage various and massive LiDAR point clouds, The American Society for Photogrammetry and Remote Sensing defined an open format, LiDAR (with ".las" file extension), for the exchange and storage of LiDAR data. This format typically consists of four components: Public Header Block, Variable-Length Records, Point Data Records, and Extended Variable-Length Records. The header block contains the metadata indicating the byte offset to the beginning of point data and the byte size of each point, which can be used to access each point efficiently [19]. This benefits in parallel and distributed algorithms for very large point cloud processing.

3.2 Chunk-based Parallel Processing

Existing point cloud processing tools, such as `Open3D` and `pypcd4`, always load a LiDAR point cloud in a single batch. However, this approach is limited in handling very large LiDAR data, as loading the whole file in a single batch can easily cause out-of-memory failures. For instance, `Open3D`, a well-known point cloud processing library, stores point coordinates, colors, and normals using `std::vector<Eigen::Vector3d>` in memory, which is memory-intensive. Loading 789-million points into memory at once requires approximately 53 GB before any further processing. Such memory demand exceeds the capacity of computers and even some workstations. Therefore, it is essential to develop a memory-efficient method to deal with massive point clouds while maintaining time efficiency.

Leveraging the structure of the LASer format, a point cloud can be randomly accessed using the following formula:

$$\text{Offset} = \text{Offset}_{\text{header}} + i \times \text{TypeSize} \tag{1}$$

where i denotes the index of the specific point to be accessed directly, and TypeSize denotes the number of bytes used to store each point (coordinates and attributes) as defined in the LASer header. This random-access property enables data access without sequential reading, allowing the LiDAR point clouds to be divided into smaller chunks to trade off between memory efficiency and processing time.

However, processing these chunks in parallel straightforwardly introduces a new challenge: each thread requires dedicated memory allocation before execution. If all chunks are loaded and processed simultaneously, it will still raise an out-of-memory error. This is because earlier tasks may not finish in time to release their allocated memory before later tasks begin to require additional memory. To address this issue, our chunk-based parallel processing approach divides tasks into multiple batches (Fig. 1) and limits the number of concurrent processes. The next batch is launched after the slowest task in the previous batch finishes.

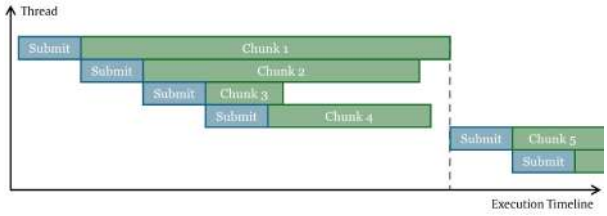

Figure 1. Illustration of chunk-based parallel processing

3.3 Axis Alignment of LiDAR point clouds

This section details the three steps of the axis alignment method that can avoid massive memory demands.

Normal estimation (Parallel computing). In the first step, we compute the normal vectors for each point. Given a point p_i, the set of neighboring points within a radius r are defined as $P = [p_1, p_2, ..., p_i, ...p_k]$. The covariance matrix of point p_i is estimated as

$$C(p_i) = \frac{1}{k} PP^T \tag{2}$$

The corresponding normal vector n_i is then obtained using Singular Value Decomposition (SVD):

$$n_i = \text{SVD}(C) \tag{3}$$

Rotation Estimation. Since the gravity direction is usually accurately provided by LiDAR scanners, the 3D rotation in axis alignment problem can be simplified to a 2D rotation estimation only around the z-axis. Specifically, by projecting each surface normal $\mathbf{n}i = (nxi, n_{yi}, n_{zi})$ onto the xy-plane, we can describe its horizontal orientation using a single rotation angle:

$$\theta_i = \arctan \frac{n_{yi}}{n_{xi}} \times \frac{180}{\pi}. \tag{4}$$

The whole angles range from $-180°$ to $180°$. We quantized the `float64` values to `uint16` integers by scaling with a factor of 100 to reduce memory usage. This is because $2^{16} = 65,536 > 360 \times 100$, all the possible angles can be represented with two-decimal precision. Therefore, three floating-point values representing normals can be compactly stored as integers without exceeding memory limits.

After chunk-based parallel processing, all rotation angles are collected in the range $[-180°, 180°)$. Since structural components such as walls, windows, and facades are often designed to fit orthogonal layouts (Fig. 3), their surface normals are distributed around 4 principal directions separated by $90°$ in the xy-plane. Leveraging this central-symmetric property, the angle distribution can be folded into the reduced range $[0°, 90°)$ to aggregate geometrically equivalent orientations, as illustrated in Fig. 2.

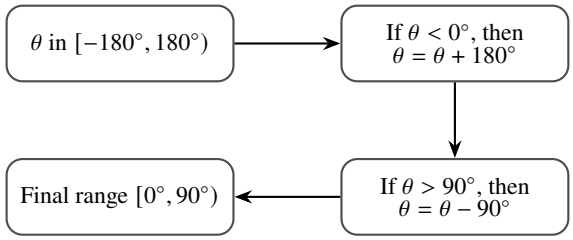

Figure 2. Range reduction from $[-180°, 180°)$ to $[0°, 90°)$

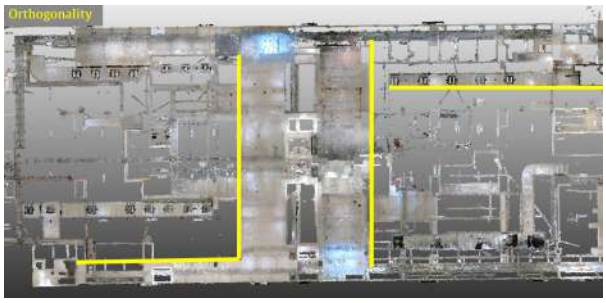

Figure 3. Illustration of the orthogonal structural components (e.g., walls) in LiDAR point clouds

Peak Direction Estimation In the final step, we extract the peak angle with the highest frequency in the range

$[0°, 90°)$. This angle denotes the dominant rotation angle between the current point cloud orientation and the global xy-axis. By rotating the point cloud around the z-axis by this estimated angle, the orthogonal building component can be parallel to the xy-axis. Note that a finer resolution in the bins of degrees can lead to more accurate peak direction estimation.

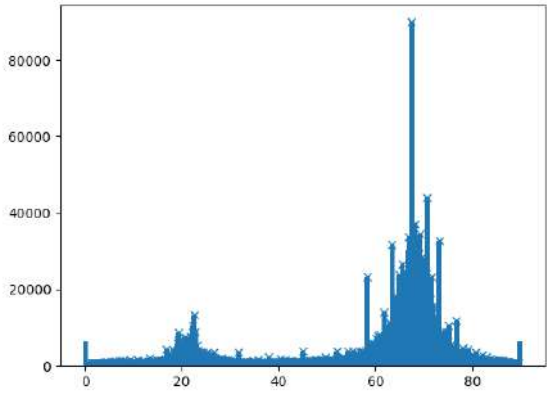

Figure 4. Degrees distribution of LiDAR point clouds normals in xy-plane

4 Experiments

4.1 Experimental Setup

To evaluate the proposed chunk-based parallel processing method for massive LiDAR point clouds, we conducted experiments on a real-world LiDAR scan dataset. The case study involves a real-world LiDAR point cloud dataset collected from the Hang Tai Road, Ma On Shan Area 86B Phase 2 construction project, a public housing development under the Hong Kong Housing Authority[1]. The scanned area covers approximately 3,392 m^2. The data were acquired using a NavVis VLX 3 system, which provides 5 mm accuracy and a capture rate of $2 \times 640,000$ points/s. The resulting LAS file is 22 GB in size and contains 789,448,875 points, with an average point density of approximately 232,738 points/m^2.

All experiments were evaluated on an HP Z2 workstation equipped with an Intel Core i7-13700K CPU, 64 GB RAM with 16 GB swap memory, running under the Windows Subsystem for Linux (WSL) environment.

Our evaluation focuses on the pipeline from reading the point cloud from disk to rotating it into alignment with the global coordinate axes. Specifically, there are three consecutive stages: (1) data loading, (2) normal estimation, and (3) axis alignment. We focus on these stages because they account for most of the memory demand throughout the point cloud preprocessing pipeline and often serve as the foundation for subsequent point cloud analyses. Therefore, evaluating these three stages can provide a comprehensive assessment of both memory efficiency and computational efficiency in very large point cloud processing.

[1]Approved Planning Brief from the Hong Kong Housing Authority

4.2 Results

We compared the proposed method with two baseline implementations, `Open3D`[2] and `pypcd4`[3], as well as a sequential chunks version of our method. As shown in Table 1, our proposed method significantly improves memory usage while maintaining computational efficiency.

Table 1. Comparison of methods for aligning the 789-million point cloud

Method	Runtime ↓	Peak RAM ↓
`Open3D` Baseline	N.A.†	>64 GB†
`pypcd4` Baseline	27.0 mins	60.5 GB
Sequential Chunks	98.6 mins	4.9 GB
Ours (this paper)	**25.6 mins**	**4.1 GB**

↓: Smaller is better; †: Failed at 23 mins.

Specifically, as illustrated in Fig. 5, `Open3D` takes around 30 minutes to load the point cloud into memory, occupying approximately 53.4 GB the system's available memory, and continues to allocate a large amount of memory, causing the process to be terminated by the system process manager. Compared with `Open3D`, which uses the `float64` format, `pypcd4` stores point coordinates in `float32` format in RAM for reducing both the memory demand (60.5 GB) and the time (27 mins) spent on memory allocation during evaluation.

The synchronization chunk processing method sequentially processes the chunk's point cloud, significantly reducing peak memory usage (Fig. 5). However, the total runtime is around 1 h 38 mins. In contrast, our proposed parallel chunk-based method achieves a trade-off between memory and time efficiency (Fig. 5). The whole process finishes within 26 minutes, which is 3.85-fold speedup compared with the synchronization chunk-based method. And the peak memory usage is also below 5 GB, considerably lower than the system's limits.

In addition to total runtime and peak RAM usage, Fig. 6 breaks the runtime into two stages, (1) data loading, (2) normal estimation and axis alignment. The results show

[2]https://github.com/isl-org/Open3D
[3]https://github.com/MapIV/pypcd4

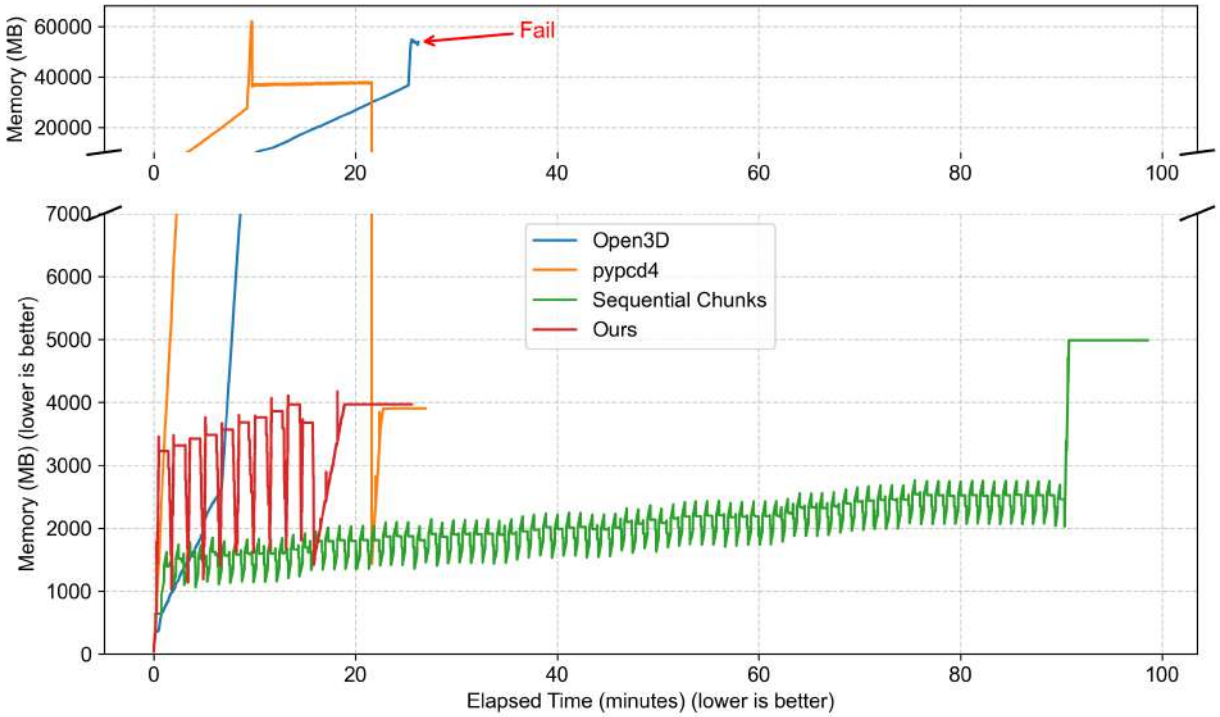

Figure 5. Memory usage comparison of `Open3D` (single batch), `pypcd4` (single batch), sequential chunk processing, and our proposed method

that most of the runtime is spent on data loading, indicating that I/O is the major bottleneck in this pipeline. Compared with the sequential chunk way, our proposed method significantly reduces the time spent on point cloud loading from disk, achieving a 4.98-fold speedup. This demonstrates that our proposed method efficiently avoids memory accumulation while maintaining time efficiency.

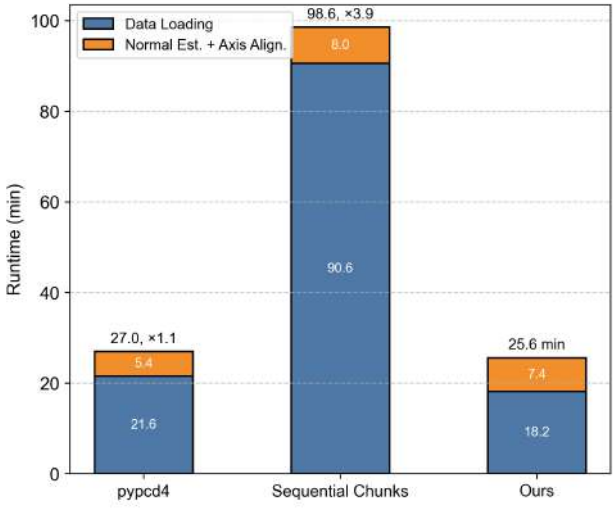

Figure 6. Runtime break down comparison of `pypcd4` (single batch), sequential chunk processing, and our proposed method

Axis Alignment Result After parallel processing, the point cloud was successfully aligned to the global 3D axis, as illustrated in Fig. 7. The red, green, and blue arrows represent the xyz-axis, respectively. The result demonstrates that our estimated dominant rotation angle enables efficient alignment of very large LiDAR point clouds to global coordinates.

Fig. 8 shows the registration result of LiDAR point clouds with the 2D drawings. As the point clouds have already been globally axis-aligned, the registration process only requires solving 2 degrees of freedom (translation along the xy-axes), significantly reducing human effort compared with full 3D registration.

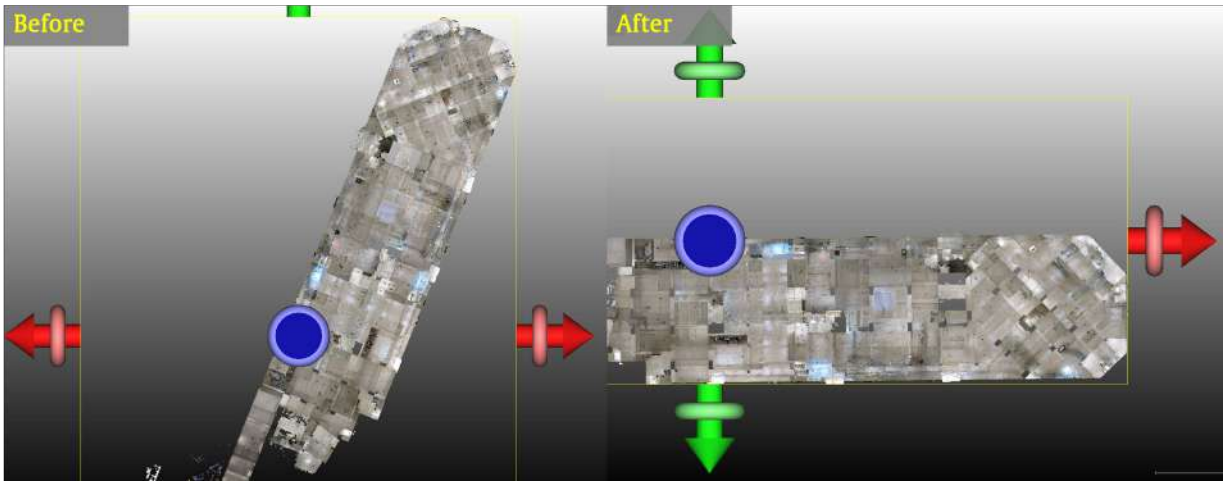

Figure 7. Axis alignment result of the LiDAR point clouds.

Figure 8. Registration result of 2D drawings with LiDAR point clouds. For better visualization, a sector along the z-axis is selected.

5 Discussion and Limitations

The experimental results demonstrated that the proposed chunk-based, orthogonality-enabled parallel computing method effectively estimates the global orientation of very large LiDAR point clouds and achieves both time and memory efficiencies. Compared with the single-batch loading approach in `pypcd4`, our method significantly reduces peak memory usage from over 60.5 GB to approximately 4.1 GB while maintaining comparable data loading time. These findings demonstrate that loading a very large point cloud into memory at once is inefficient when memory is limited. In contrast, aggregating local geometric cues from chunks without full memory access to the full point cloud can be a more efficient way to estimate global orientation and preprocess.

However, several limitations still exist. First, the proposed method currently supports only the LASer file format, although ".PLY" and ".PCD" also have random-access properties. Second, the hyperparameters, such as the chunk size and the number of concurrent processes, are set manually, leading to underutilization of RAM and CPU resources. Last but not least, the method was not tested across diverse building and construction scenes. More importantly, the assumption of dominant orthogonality may not fit for all buildings (e.g., curved facades and non-Manhattan layouts). In such cases, the degree distribution of surface normals may have multiple peaks, potentially degrading the global orientation estimation and failing the axis alignment. In addition, this study does not provide experiments on how chunk size affects runtime and memory usage, which should be further explored in future work.

6 Conclusion

This study presents a chunk-based, orthogonality-enabled method for automatically aligning the global axes of very large LiDAR point clouds. By leveraging the random-access property of the LASer file and controlling parallel concurrency, our proposed method efficiently achieves massive point cloud loading, normal estimation, and global axis alignment on very large LiDAR point clouds on computers. Experiments are conducted on a 22 GB real-world LASer file. The experimental results demonstrated that our method drastically reduces memory consumption and achieves a 3.85-fold speedup compared with the sequential chunk-based approach. The parallel computing mechanisms in this axis alignment method can also benefit other applications of LiDAR point clouds, including learning-based point cloud analysis, point cloud-to-CAD/BIM registration, change detection, and construction progress monitoring.

For future work, one can (1) extend the support for point cloud exchange formats, such as `pypcd4`'s ".PCD" and Stanford polygon ".PLY", (2) analyze the influence of chunk size and concurrency on the memory and CPU utilization, and (3) evaluate the method on a wider range of very large scanning datasets.

Acknowledgment

The data used in this study were collected and provided by Geosys Hong Kong Limited. The work described in this study was supported by the Hong Kong Research Grants Council (No. 17201325) and, in part, by the Hong Kong Innovation and Technology Commission (No. ITP/004/23LP).

References

[1] Mingyu Zhang, Yushu Yang, Shuai Han, Heng Li, Dongliang Han, Xiaotong Yang, and Nan Guo. Lidar-based framework for accurate positioning and robust tracking of multiple construction workers. *Journal of Computing in Civil Engineering*, 39(3): 04025027, 2025.

[2] Paul Sestras, Gheorghe Badea, Ana Cornelia Badea, Tudor Salagean, Sanda Roșca, Shuraik Kader, and Fabio Remondino. Land surveying with uav photogrammetry and lidar for optimal building planning. *Automation in Construction*, 173:106092, 2025.

[3] Wenyao Liu, Jinhua Chen, Zemin Lyu, Rui Feng, Tong Hu, and Lu Deng. Automatic tile position and orientation detection combining deep-learning and rule-based computer vision algorithms. *Automation in Construction*, 171:106001, 2025.

[4] Andrew Yarovoi and Yong Kwon Cho. Review of simultaneous localization and mapping (slam) for construction robotics applications. *Automation in Construction*, 162:105344, 2024.

[5] Feng Li, Wenzhong Shi, Yunlin Tu, and Hua Zhang. Automated methods for indoor point cloud preprocessing: Coordinate frame reorientation and building exterior removal. *Journal of Building Engineering*, 76:107270, 2023.

[6] Lin Bie, Shouan Pan, Siqi Li, Yining Zhao, and Yue Gao. Graphi2p: Image-to-point cloud registration with exploring pattern of correspondence via graph learning. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 22161–22171, 2025.

[7] Ke You, Ke Chen, and Fan Xue. Automated scan-to-bim for construction digital transformation: Conceptual framework, processing methods and best-

practice guidelines. *Automation in Construction*, 182:106755, 2026.

[8] Iris De Gélis, Thomas Corpetti, and Sébastien Lefèvre. Change detection needs change information: Improving deep 3-d point cloud change detection. *IEEE Transactions on Geoscience and Remote Sensing*, 62:1–10, 2024.

[9] Dong Liang, Longyong Wu, Meng Sun, Ruibo Hu, Lingming Kong, Yipeng Pan, and Fan Xue. Transfer learning from building information model-based synthetic data for three-dimensional module detection in point clouds of modular-integrated construction hoisting. *Engineering Applications of Artificial Intelligence*, 164:113243, 2026.

[10] Maxim Kolodiazhnyi, Anna Vorontsova, Anton Konushin, and Danila Rukhovich. Oneformer3d: One transformer for unified point cloud segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20943–20953, 2024.

[11] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018.

[12] Yijie Wu, Fan Xue, Maosu Li, and Sou-Han Chen. A novel building section skeleton for compact 3d reconstruction from point clouds: A study of high-density urban scenes. *ISPRS Journal of Photogrammetry and Remote Sensing*, 209:85–100, 2024.

[13] Dong Liang, Sou-Han Chen, Zhe Chen, Yijie Wu, Louis YL Chu, and Fan Xue. 4d point cloud-based spatial-temporal semantic registration for monitoring mobile crane construction activities. *Automation in Construction*, 165:105576, 2024.

[14] Fan Xue, Weisheng Lu, Christopher J Webster, and Ke Chen. A derivative-free optimization-based approach for detecting architectural symmetries from 3d point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, 148:32–40, 2019.

[15] Fan Xue, Weisheng Lu, Ke Chen, and Anna Zetkulic. From semantic segmentation to semantic registration: derivative-free optimization–based approach for automatic generation of semantically rich as-built building information models from 3d point clouds. *Journal of Computing in Civil Engineering*, 33(4): 04019024, 2019.

[16] Yijie Wu, Fan Xue, Liangliang Nan, Longyong Wu, Jantien Stoter, and Anthony GO Yeh. Morphcut: an efficient convex decomposition method of 3d building models for urban morphological analytics. *International Journal of Geographical Information Science*, pages 1–23, 2025.

[17] Xiaolu Zeng, Yang Hu, Xiaopeng Yang, Zixiang Yin, and Shichao Zhong. A multi-source fusion system for through-wall radar compensation using lidar and slam-based 3d reconstruction. *IEEE Sensors Journal*, 2025.

[18] Chunran Zheng, Wei Xu, Zuhao Zou, Tong Hua, Chongjian Yuan, Dongjiao He, Bingyang Zhou, Zheng Liu, Jiarong Lin, Fangcheng Zhu, et al. Fast-livo2: Fast, direct lidar-inertial-visual odometry. *IEEE Transactions on Robotics*, 2024.

[19] Andre Samberg. An implementation of the asprs las standard. In *ISPRS Workshop on Laser Scanning and SilviLaser*, pages 363–372, 2007.