Real-time 3D Space Reconstruction for OT Metaverse as Interactive Virtual Site

Keiichi MITANI[1], **Kazuyuki TAJIMA**[1], **and Yusuke NAKAMURA**[1]

keiichi.mitani.jj@hitachi.com
[1]Center for Digital Services – Instrumentation Hitachi, Ltd. Research & Development Group

Keywords: Real-time, Point cloud, Surface reconstruction, Metaverse, Boolean

ABSTRACT

We developed a platform for real-time 3D digitization of a work site and a method for high-quality, fast 3D-geometry modeling are proposed. It was confirmed that the platform works in real time, less than the target specification of 1 second, and the proposed modeling method is faster and robust than conventional methods.

1 INTRODUCTION

Many studies have tried to create a so-called “cyber physical system” (CPS), which simulates any phenomenon in cyberspace by digitizing physical information in the real world and feeding back the results to the real world [1]. In recent years, a CPS as a “metaverse,” in to which people can enter and interact with each other, has been actively researched [2]. Distance-measuring technology has also advanced, and low-cost distance-measuring sensors have become abundant. For example, a ranging sensor can digitize and reconstruct in 3D the on-site space that exists within the area of its measurement range.

Supposing support of on-site work remotely as a use case, we aim to enable remote skilled engineers to provide support to on-site workers in a natural manner. In particular, we want to improve the quality of work support by providing a realistic “metaverse space” of the work site. The space is generated by using multiple ranging sensors installed at the site and modeling its main elements as 3D geometry data, which allows remote users to interact with objects at the site via the metaverse space. However, to allow users to interact with each other via objects, real-time modeling is necessary. Moreover, when the user is immersed in the metaverse space and approaching objects modelled as 3D geometry data, it is necessary to maintain a resolution that allows the user to view the object data sufficiently. In this study, we built a platform for real-time 3D digitization of the site and devised a method for improving reality and generating 3D-geometry data structures at high speed.

2 REAL-TIME 3D RECONSTRUCTION OF A WORK SITE

2.1 Platform Configuration

The developed platform converts the site space into 3D-point-cloud data, which can be monitored from remote applications by a user at a free viewpoint. The main functional blocks that operate on the platform are shown in Figure 1.

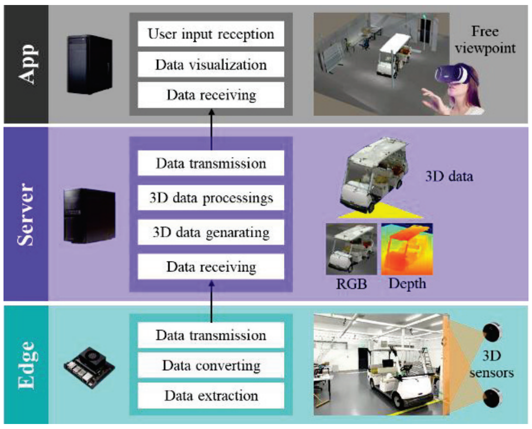

Fig. 1: Functional blocks of the platform

The platform is divided into three layers: “edge” extracts data frames from multiple ranging sensors installed in the site space and converts them into color and depth images; “server” converts color and depth images into 3D-point-cloud data and stores them; and “app” generates free-viewpoint images for the user to view by referring to the 3D-point-cloud data within the necessary range. The edge, server, and app environments were built separately, and data can be exchanged between them via the WebSocket protocol, which is lightweight and allow users to define their own formats. In the 3D-point-cloud data maintained by the server layer, dynamic regions change from moment to moment, and static regions remain unchanged for long periods of time. As for the static regions, it is inefficient from both practical and bandwidth-intensive standpoints to transmit and update 3D-point-cloud data constantly; accordingly, the data is scanned in advance, compressed, and stored in external storage.

The hardware used in the experiment is listed in Table 1. Connected to the edge layer, Intel RealSense LiDAR cameras (L515) were used as the sensors. Their resolution is 1280 × 720 for color images and 1024 × 768 for depth images.

Table 1. Hardware configuration

Layer	CPU
Edge	6 cores, 8 GB RAM
Server	8 cores, 32 GB RAM
App	10 cores, 64 GB RAM

2.2 3D Dynamic Surface Reconstruction

Views of the subject in the application seen from (a) far and

(b) near are shown in Figure 2. Owing to the spatially sparse data structure of 3D-point-cloud data, the object appears thinner and less visible as the viewer moves from (a) to (b). To solve this problem, two simple and lightweight processing methods are available: "voxel filling," which increases the size of the points when drawing to avoid thinning of the subject by pseudo-spatial interpolation, and "surface reconstruction," which connects points with polygonal surfaces to form discrete surfaces and interpolates space by filling the surfaces with the colors of the vertices.

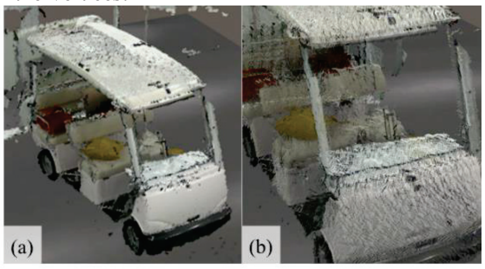

Fig. 2: Views of the subject on the application seen from (a) far and (b) near

The conventional procedure for 3D surface reconstruction consists of two steps: (i) reducing noise and simplifying vertices of input 3D-point-cloud data and (ii) generating a mesh by connecting adjacent vertices. However, this procedure targets static structures, and it must balance a tradeoff between speed and robustness. For example, Delaunay triangulation [3], with complexity of $O(N^{3/2})$ (where N is the number of vertices in the point cloud), is fast, but it is sensitive to noise and requires resources to be devoted to noise reduction. Conversely, Poisson surface reconstruction [4] is robust, but it has complexity of $O(N^3)$, so it not suitable for real-time purposes. Furthermore, from a real-time perspective, the simplification of vertices precedes the mesh generation, and that results in loss of features related to shape such as normals and curvature. Conversely, decimation is applied after the mesh is generated, but without simplification of the decimation process, mesh generation is resource intensive.

In this study, we solve the above both problems by sequentially combination of the fast Poisson disk sampling [5] and 2D Delaunay triangulation. Firstly, we use 2D depth images instead of 3D-point-cloud to input the method thus it is faster than the conventional method at the point of spatial dimension. Our proposed surface-reconstruction method is shown schematically in Figure 3. Fast Poisson disk sampling, with complexity of O(N), is a means of "exclusive sampling"; that is, the next trial for the sampled points is performed within a radius r to $2r$. Varying the value of r according to the depth value enables downsampling in real 3D space. In addition, normals and curvature can be calculated from the surrounding pixel values and added to the sampling to preserve shape characteristics. And stochastic and exclusive sampling is used for noise reduction. 2D Delaunay triangulation generates surfaces so that the distribution of triangulation ratio, which is O(N) in the case of 2D space, is uniform. Moreover, it is robust because noise is removed in the previous processing (fast Poisson disk sampling). The methods used in the conventional procedures for surface reconstruction are compared with those used in this study in Table 2.

From the above, our proposed the newly surface-reconstruction method is compatible with real-time and robustness. Here, as the first step, we focus on slow-moving use cases such as equipment inspection, maintenance, construction, and mining. We set the target specification less than 1 second of update frequency (> 1 fps) for real-time 3D digitization to adequately track details of an operation in that use cases.

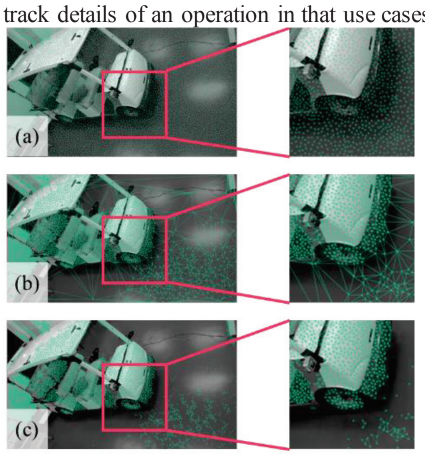

Fig. 3: Schematic of the proposed surface-reconstruction method: (a) fast Poisson disk sampling, (b) 2D Delaunay triangulation, and (c) long-edge removal

Table 2: Comparison of conventional and proposed surface-reconstruction methods: (a) voxel down sampling, (b) decimation, (c) 3D Delaunay triangulation, and (d) Poisson surface reconstruction.

Method		Real-time	Robustness
(a)	(c)	✓	
	(d)		✓
(b)	(c)	✓	
	(d)		✓
Ours		✓	✓

2.3 3D Dynamic Geometry Reconstruction

The conventional procedure for 3D surface reconstruction is to merge multiple input 3D-point-cloud data, reduce noise and simplify vertices, and connect adjacent vertices to generate a closed mesh. The difference between 3D dynamic geometry described in the previous section is that multiple 3D-point-cloud data are to be input. However, the size of the integrated 3D-point-cloud data is huge, so it takes time to process each point. The conventional method is thus unsuitable for real-time purposes.

Accordingly, in this study, the proposed reconstruction method described in the previous section is implemented for each sensor and processed in parallel, and the surface data reconstructed for each 3D surface is integrated at the end. However, two problems emerge: (i) the need to perform dynamic and fast 3D Boolean operations on static data and (ii) the need to determine intersections in open-surface structures robustly. To solve these two problems, the following two approaches were tried: (i) region determination by applying a signed distance function (SDF) and surface stitching focusing only on intersection regions and (ii) incorporating a mesh-arrangement method [6] that allows non-diverse structures. As for the first approach, it has been reported that a triangular mesh structure can be used to compute the SDF at high speed [7]. A schematic of the proposed method (3D geometry reconstruction) is shown in Figure 4. The SDF representation can simply extract out/in bounding data like (b) by its sign because all points have a signed distance from their position to the boundary. Therefore the bounding data like (c), equivalent to an intersection, is also simply extracted, and its computational complexity of that is O(1); only accessing of its reference data.

From the above, our novel method can process rapidly even if the data size becomes larger by increasing of input data, edge sensors, and so.

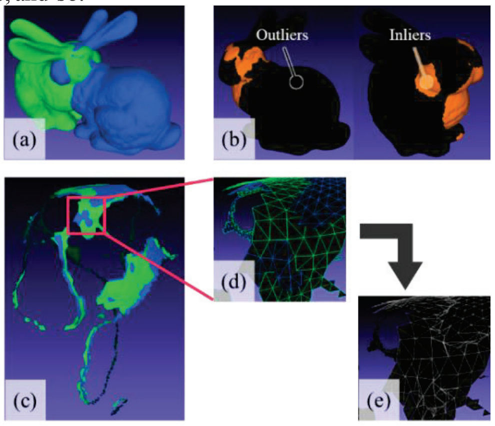

Fig. 4: Schematic of the proposed method for 3D geometry reconstruction: (a) input data, (b) inner and outer regions, (c) intersection boundaries determined by using SDF, (d) partially enlarged intersection boundary, and (e) "stitched" intersection boundary.

3 EXPERIMENTAL RESULTS AND DISCUSION

3.1 Performances of the Platform

Data-update frequency and data-transmission rate—when the on-site space is converted to 3D digital data and visualized on the remote user application—are plotted in Figure 5. Voxel downsampling was used to thin out the data at equal intervals, and the aforementioned frequency and rate were measured while the total amount of 3D-point-cloud data was varied. A real-time development platform, described in 2.1, was used to visualize the 3D-point-cloud data. Data was transmitted via an in-house network environment. Frame rate of 3 fps or more was ensured, and it was confirmed that the on-site information could be digitized in real time (> 1 fps).

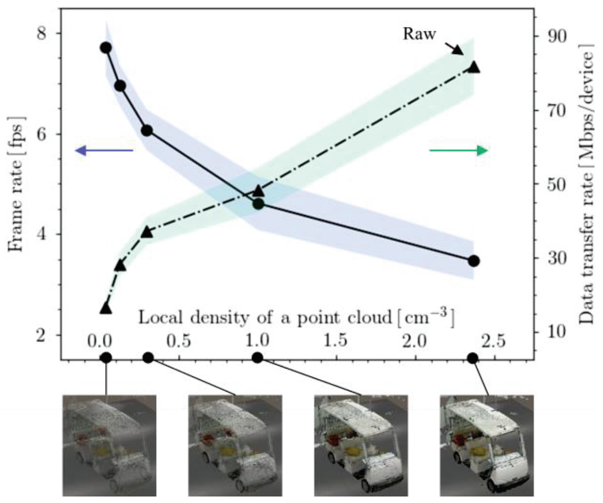

Fig. 5: Trade-off between frame rate and appearance quality (data size). The circles plot and triangles plot respectively indicate frame rate and data-transfer rate. The filled regions indicate standard deviation. The pictures aligned under the x axis are object views at corresponding local density of the point cloud.

3.2 View of the point-cloud subject as seen via the application

Snapshots of the 3D data of the work site visualized on the remote user's application and observed from longer and shorter distances from the subject (golf cart) on the application are shown in Figure 6. On the left of the figure, the point-cloud data of the subject is viewed from (a) longer and (b) shorter distance. On the right, the mesh data of the subject is viewed from (c) longer and (d) shorter distance.

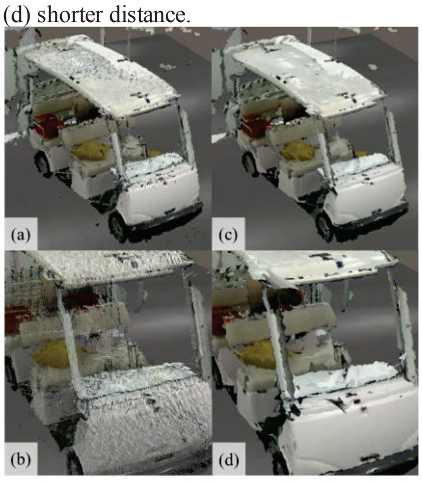

Fig. 6: Views of the point-cloud subject as seen via the application seen from (a) longer and (b) shorter distance and views of the meshed subject (golf cart) seen from (c) longer and (d) shorter distance

Although no significant difference can be seen between (a) and (c), a clear difference can be seen between (b) and (d). In other words, it was confirmed that the surface reconstruction using the proposed method improves visibility at close proximity. It was also confirmed that frame rate of 2.25 fps guarantees real-time performance (> 1 fps).

3.3 Performance of 3D Geometry Reconstruction

Processing speed and robustness in the case of inputting two meshes, stitching them together (Union processing), and outputting them as a single mesh were evaluated. The surface (test) data used for the evaluation are shown in Figure 7.

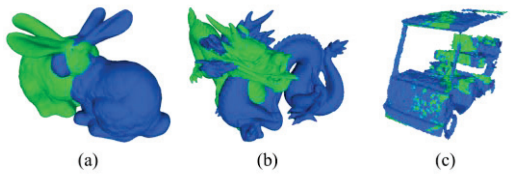

Fig. 7: Test data used for performance profiling: the opposed-overlapped (a) bunnies and (b) dragons from Stanford 3D data and (c) aligned golf cart.

In particular, "small bunny" data (69,451 faces/34,834 vertices) and "large dragon" data (871,414 faces/437,645 vertices) were taken from the Stanford dataset [8], namely, a noise-free dataset widely used for validation, and as actual data, "cart" data (27,458 and 30,110 faces/14,652 and 16,036 vertices) were generated by scanning the on-site subject (golf cart) with the sensors by the method described in Section 2.2. Other open, general-purpose 3D Boolean libraries, namely, CGAL [9], libgl [10], and Cork [11], were used for comparison with the proposed method.

The results of the evaluation are listed in Table 1. When the dragon data was processed with CGAL, the processing was not complete even after more than one hour, and the output could not be confirmed. When the cart data was processed with CGAL and libgl, processing time could not be measured because the original structure was greatly disrupted. As for the "bunny" data, the proposed method ("Ours") took longer than CGAL, but the difference in processing times was within 0.1 s.

Table 3: Comparison of processing times of proposed and conventional reconstruction methods performing zippering

Subject	Process time [s]		
	Bunny	Dragon	Cart
CGAL	**0.331**	> 3600	-
libigl	2.057	19.976	-
Cork	2.351	42.755	1.798
Ours	0.423	**5.440**	**0.227**

4 CONCLUDING REMARKS

A platform for real-time 3D digitization of a work site was developed, and a method for improving reality and generating 3D-geometry data structures at high speed was proposed.

(1) We established a platform for generating free viewpoints on a user application at a remote location that based on 3D data measurement taken by multiple ranging sensors installed at the site. The end-to-end data update frequency is about 3 fps or higher, and real-time (> 1 fps) operation of the platform was confirmed.

(2) We proposed the novel surface-reconstruction method to improve visibility on the user application that generates 3D surfaces from 2D depth images by the fast Poisson disk sampling and 2D Delaunay triangulation. Surfaces that were robust to noise could be generated, and improved visibility at close proximity was confirmed. The data-update frequency of 2.25 fps confirms the real-time (> 1 fps) nature of the data.

(3) We also proposed the novel method for dynamic geometry reconstruction to stitch together surface data from multiple viewpoints and model them as 3D geometry data, an SDF was combined with a robust mesh-arrangement method, and 3D Boolean operations were performed on static data dynamically at high speed. It was confirmed that the proposed method is robust, and its processing operation is fast compared to the existing general-purpose 3D Boolean libraries. Furthermore, in the experiment using the actual data including noises, the processing time of 0.227 s is 7 times faster or more, and confirms the target specification of less than 1 s.

These technologies (i.e., the developed platform and proposed reconstruction method) will enable real-time interaction with objects in the field from metaverse space and could be widely applied to industrial applications. Our future work is acceleration of processing to expand use cases.

References

[1] Y. Okumura, The Journal of Science Policy and Research Management, vol. 32, no. 3, pp. 251–265, 2017.
[2] H. Ning, et al., CoRR abs/2111.09673, 2021.
[3] Jonathan Shewchuk Siu-Wing Cheng, Tamal K. Dey, Delaunay Mesh Generation, Chapman and Hall/CRC, 2012.
[4] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe, SGP '06: Proceedings of the fourth Eurographics symposium on Geometry processing, 2006.
[5] Robert Bridson, SIGGRAPH '07: ACM SIGGRAPH 2007 sketches, 2007.
[6] Gianmarco Cherchi, Marco Livesu, Riccardo Scateni, and Marco Attene, ACM Trans. Graph., vol. 39, no. 16, 2020.
[7] J. A. Bærentzen and H. Aanæs, 2002.
[8] G. Turk and M. Levoy, in Proceedings of the 21st annual conference on Computer graphics and interactive techniques, pp. 311–318, 1994.
[9] Cgal, Computational Geometry Algorithms Library. http://www.cgal.org.
[10] Alec Jacobson, Daniele Panozzo, et al., libigl: A simple C++ geometry processing library. https://libigl.github.io/.
[11] G. Bernstein, Cork Boolean library. https://github.com/gilbo/.