UNB EST. 1785 UNIVERSITY OF NEW BRUNSWICK | Off-site Construction Research Centre

CSCE CGC 1887 Canadian Society for Civil Engineering

**Transforming Construction with Reality Capture Technologies:
The Digital Reality of Tomorrow**

August 23-25, 2022, Fredericton, New Brunswick, Canada

ROBOT-BASED REAL-TIME POINT CLOUD DIGITAL TWIN MODELING IN AUGMENTED REALITY

You, Hengxu[1], Xu, Fang[2], and Du, Eric[3,*]
[1] Informatics, Cobots and Intelligent Construction (ICIC) Lab, Department of Civil and Coastal Engineering, University of Florida, United States, you.h@ufl.edu
[2] Informatics, Cobots and Intelligent Construction (ICIC) Lab, Department of Civil and Coastal Engineering, University of Florida, United States, xufang@ufl.edu
[3] Informatics, Cobots and Intelligent Construction (ICIC) Lab, Department of Civil and Coastal Engineering, University of Florida, United States Eric.du@essie.ufl.edu. Corresponding Author.

Abstract: Augmented Reality (AR) can be used as an intuitive user interface of the reality capture models for structural inspection, emergency response and teleoperations. For a seamless AR user experience, reality capture models usually need to be collected, processed, and rendered in a real-time manner. Nonetheless, most scanning technologies require a prolonged in-situ time on the site for data collection. And due to the excessive amount of data to be transferred and processed, it is difficult to achieve real-time scene reproduction. It is also challenging to render high-fidelity captured scenes via most commercial AR devices that do not have powerful graphic processing units for rapid renderings, especially when there are dynamic objects. This paper proposes an AR-based real-time 3D scene rendering method solely based on low-resolution mobile LiDAR. Velodyne-16 is used to collect dynamic point cloud data given its high stability in mobile scanning and low price compared to other high-resolution LiDAR devices. To maintain an economic data transmission to an AR headset, the obtained points message only includes the intensity and three-dimensional coordinates, relatively sparse. To overcome the defect of data sparsity, a scene texture information enrichment algorithm is implemented based on SLAM. Using IMU data and the LOAM algorithm, the scanning information of each angle position is transferred into a coordinate system with the initial position as the origin, and the supplementary scene reconstruction result containing denser texture information is obtained. To eliminate duplicate or similar scan points, voxel down-sampling is used after merging all scans, and the down-sampled point cloud will be used as a fixed background. Finally, the VFX and Shader processes are applied to render textural and physical information through ROS-Unity. The real-time scene scanning results can be displayed with a satisfactory quality via HoloLens 2 with dynamic objects. The method is expected to contribute to rapid 3D scene reconstruction based on economic sensing and processing methods, as well as LiDAR integration with extended reality technologies.

Keywords: Augmented Reality, 3D indoor reconstruction; LiDAR; Digital Twins; SLAM

1. INTRODUCTION

The continuous evolution of Augmented Reality (AR) technologies has shown great potential to be an intelligent assistant for many construction tasks (Schiavi, Havard et al. 2022). By integrating complex visual information and reconstructing the jobsite three-dimensional (3D) mapping, AR enhances the workers' perception of the surrounding environment and enriches the users' awareness of potential hazards(Kim,

Kim et al. 2021). The major challenge for visualization in AR is to provide accurate and rapid 3D reconstruction results based on vision sensors and limit external disturbances such as lightning changes or device vibrations (Hua, Nan et al. 2019). The present AR rendering methods mainly rely on the environmental understanding functions of AR devices, such as HoloLens (Bekele and Champion 2019). The default sensors equipped with AR devices, including RGB-D cameras and distance sensors, are used to support streamlined machine learning algorithms for environmental modeling and registration (Karkus, Cai et al. 2021). Although it works for most scenarios, AR-equipped sensing and modeling are subject to several obvious limitations, such as the sensitivity to ambient lighting, limits of memory for storing the mesh models, and the computing efficiency for feature recognition and registration when moving in a bigger space (Rodrigues, Nagamura et al. 2019) (Xiao and Lifeng 2014).

With the vigorous development of sensor technologies, different types of sensors emerge to provide external and secondary point cloud data for AR devices. The sensors can be roughly grouped into two types, including LiDAR (light detection and ranging) and RGB-D cameras. RGB-D camera can provide sufficient visual details and capture pixel-wise information for scene reconstruction and indoor segmentation and detection (Song, Lichtenberg et al. 2015), but the imagery data can be easily influenced by lighting conditions and lack the ability to support feature extraction without significant spatial structures (Jonasson, Ramos Pinto et al. 2021). LiDAR, on the other hand, can provide spatial information in a larger range with high stability. However, the obtained data points of most mobile LiDAR sensors are sparsely distributed with disorderliness causing difficulties in textural information collection and object shape recognition (Yang, Chao et al. 2013). Although the lack of detailed information could be solved by using LiDAR points as the backbone and fusing with other visual inputs (Vora, Lang et al. 2020), the fusion process requires additional computational time and requires an accurate calibration process. Furthermore, it is still unclear how to optimize the rendering results of the raw LiDAR in most commercial AR devices, as the data processing pipelines are different between the LiDAR platform (usually ROS) and AR platform (usually Unity), causing unsatisfactory results in AR devices, such as loss of texture information or shading.

In this paper, we propose a novel indoor scene reconstruction method by utilizing only low-resolution LiDAR for high-fidelity 3D model rendering via light AR HMD. We take LOAM (Zhang and Singh 2014) as our basic SLAM structure which is originally used for large-scale outdoor environment reconstruction. We modify the transformation matrices estimation process to augment the scanning result of the first point cloud frame by the following ones in a single reconstruction process. The scanning results from different viewing angles will be combined to build a dense map for a given scene. In addition, we add the real-time scanning result onto the completed map so that the moving objects such as humans or other machines could be highlighted to provide dynamic information. Furthermore, we build the connection bridge between ROS and Unity so both the reconstructed map and real-time scanning results could be visualized in the AR system with a near-real-time frequency. To remove the redundant points from stacked frames, we also applied a simple down-sampled method to filter the map and compare the final visual results with different sets of parameters. We also utilized a VisualEffect (VFX) shading method to optimize the rendering results in HoloLens 2.

2. RELATED WORK ABOUT 3D RECONSTRUCTION

The 3D reconstruction sensors can be roughly grouped into LiDAR and RGB-D cameras regarding different data collecting methods. For image-based methods, monocular or stereo images are the generally used inputs for point cloud data collection. The textual information from single images will be captured and used to extract corresponding features based on computer vision algorithms. Then the extracted features from consequential images will be matched to compute the transformation of camera pose and to integrate the depth and RGB-D information to generate point clouds. The detailed texture information captured by RGB-D camera could be applied to solve different types of indoor vision tasks such as scene completion (Dai, Diller et al. 2020), transparent object pose estimation (Sajjan, Moore et al. 2020) and oriented object detection (Shi, Guo et al. 2020). Conversely, the drawbacks of RGB-D input are obvious: the images are

easily influenced by lighting conditions and lack the ability to support feature extraction without significant spatial structures. Besides, RGB-D images provide sufficient texture information for feature extraction so they could be combined with SLAM for indoor localization and mapping in the field of computer vision and robotics. However, the captured data is easily influenced by external disturbance which will influence the feature point registration process of consequent frames and lead to poor mapping results.

LiDAR could provide stable scanning results. The sensor emits laser rings and gathers the reflectance to detect the distance from the object's surface so it is invariant to lighting changes or disturbances. Thus, LiDAR sensors are widely used in self-driving (Geiger, Lenz et al. 2012, Caesar, Bankiti et al. 2020, Sun, Kretzschmar et al. 2020), remote sensing (Zhu, Gehrung et al. 2020) and large-scale mapping (Tan, Qin et al. 2020). The high quality and stability of the acquired data ensure the improvement of accuracies for different vision tasks such as object detection (Jiang, Zhang et al. 2022), motion tracking (Huang and Hao 2021) and path planning (Bolourian and Hammad 2020). However, the LiDAR-based methods still suffer of lacking in sufficient reflected data points to extract features for detection from a scanning frame. Recently, many state-of-the-art algorithms aim to take advantage of the spatial information from the LiDAR point cloud from different points of view, such as using voxel representations with different scales to concatenate multi-scale features (Zhou and Tuzel 2018) or combining point features with their corresponding voxelized features (Shi, Guo et al. 2020). Despite the fact the usage of multi-scale features could enhance the performance of either detection or 3D reconstruction, the key problem has remained unsolved that the input data source of LiDAR is too sparse to support complex environment completion, especially for the indoor environment which contains detailed small-scales spatial information.

3. SYSTEM OVERVIEW

In this section, we describe the pipeline of our proposed LiDAR-SLAM 3D indoor dense reconstruction system for AR supplementary scene reconstruction. We introduce some pre-defined parameters that will be used in the following discussion for convenience. We also list the hardware information and algorithm description as an overview of our method.

3.1 Task a Parameter Definition

Our method aims to build a dense 3D scanning map for an indoor environment based on a single LiDAR that could capture more detailed spatial information with real-time operation speed. We use $PC = \{(p, l)|p \in R^3, l \in R\}$ to denote a single point cloud, where *p* stands for the coordinates of a returned point and *I* stands for the corresponding intensity. $S = \big(O, (H_x, H_y, H_z)\big)$ denotes the LiDAR coordinate system where $O, H_x, H_y, H_z \in R^3$ with *O* denotes the origin coordinates of the current LiDAR system in the world coordinate of the scanning site and (H_x, H_y, H_z) denotes the direction of x-, y- and z-axis. $T \in R^{4*4}$ refers to the transformation matrix between two point clouds or origin coordinates. At timestamp i, S_{i-1}, PC_{i-1} and S_i, PC_i denote the adjacent LiDAR coordinates and point clouds then the transformation from $i-1$ to i could be represented as $PC_i = T_{i-1}^{i} \cdot PC_{i-1}$ or $O_i = T_{i-1}^{i} \cdot O_{i-1}$.

3.2 Hardware Description

Our method is designed to use a single low-resolution mobile LiDAR scanner to build a dense 3D reconstruction map for indoor scenes. We use Velodyne VLP-16 LiDAR to collect point cloud and the method could also be extended and implemented based on the other types of LiDAR sensors. For convenience, we use V16 to denote the LiDAR sensor in the following discussion. V16 has 30 degrees vertical scanning range from -15° to 15° with 16 horizontal laser rings r_i, where *i* denotes the index of rings following the top-to-down order88. Each ring includes 360-degree scanning points with a rotation speed ranging from 300 RPM (revolutions per minute) to 1200 RPM. In our implementation, we simply set the rotation speed as 600 RPM. The laser firing time is 55.296μs with 0.199° azimuth resolution. Figure 1 shows the top and front views of the VL16 scanner. To be consistent with the Velodyne manual, we use α to represent the vertical angle and ω to represent the horizontal rotation angle. The coordinates of receiving points are assigned in a right-hand system and approximately 28800 points are collected for each scanning

recycle. As for the platform, we use a quadrupedal robot for mobile LiDAR scanning. We programmed a unique locomotion pattern for the robot as it moves a certain angle vertically for filling the sparse LiDAR data. It will be discussed in detail in the following sections.

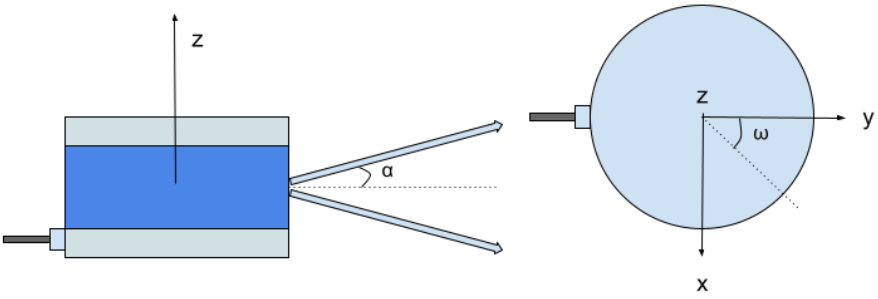

Figure 1: Front view (on the left) and Top view (on the right) of Velodyne VLP 16.

3.2 Software Setup

Our developed method can be divided into two parts: LiDAR-based LOAM and reality capture visualization. For the first part, we build the modified LOAM algorithm with ROS melodic on Ubuntu 18.04 with an Intel i7-8750H CPU and an NVIDIA GeForce GTX 1050 Ti GPU. For the latter part, we build the data transmission bridge with ROS Sharp between the Ubuntu laptop and a Windows PC with an Intel i9-11900F CPU and an NVIDIA GeForce GTX 3080 GPU. The reconstructed map and dynamic scanning results are visualized in Unity with version 2020.3.25. In Figure 2, we illustrate the overall structure of our 3D reconstruction system.

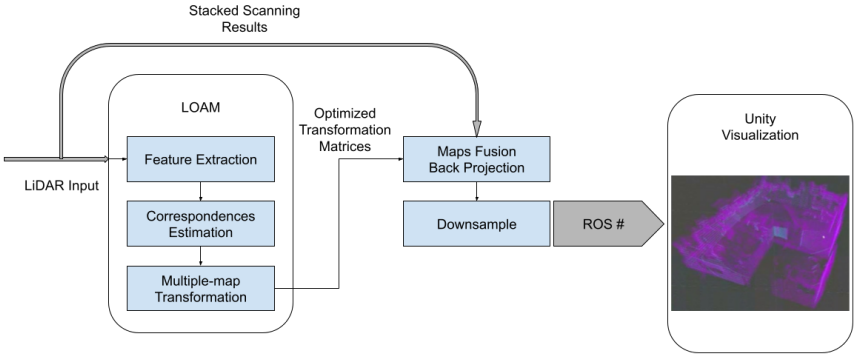

Figure 2: The overview of our 3D reconstruction and reality capture pipeline.

4. METHOD

4.1 Feature Point Selection

Our reconstruction SLAM method is built based on LOAM (Zhang and Singh 2014). The overall process includes three major steps: feature point selection, correspondences estimation and multi-map transformation. For a given point cloud PC_t, the points on edges and planar surfaces are selected as feature points, denoted as E_t and Pl_t, respectively. According to Section 3.2, PC_t consists of 16 calibrated ring-shaped scanning point sets denoted as $r_k, k = \{1,2,\dots,16\}$. The points in a single ring are obtained in clockwise order. Let $p^t_{(i,k)}$ be a single point from r_k in local coordinate at timestamp t and PC_s be the set of continuous points of $p^t_{(i,k)}$ that is equally distributed on both sides of $p^t_{(i,k)}$. Then the sharpness of the local area around $p^t_{(i,k)}$ could be evaluated based on the following equation:

$$s = \frac{1}{|PC_s| \cdot \|p_i^k\|} \left\| \sum_{j \in PC_s, j \neq i} (p_i^k - p_j^k) \right\|. \quad (1)$$

Thus, for each point $p_{(i,k)}^t$ there is a corresponding sharpness score $s_{(i,k)}^t$. All the points are filtered based on s values. The points with the highest s scores are saved as edge points and the points with the lowest ones will be stored as planar points. According to (Zhang and Singh 2014), a single ring is divided into 4 subregions on the x-y plane and each subregion includes 2 edge points and 4 planar points. In other words, the points in a single subregion will be firstly sorted in a descending order based on their s values and the top 2 and bottom 4 points will be treated as the region's feature points. In addition, two types of insufficient feature points will be removed to ensure high evaluation quality for the downstream process. Firstly, the points on planes that are parallel to the ring plane will be dropped. Secondly, the points on the occluded planar will be ignored. Figure 2 shows the second situation. Planar A is half occluded by Planar B and point a is close to Planer B. Thus, even if point *a* is a planar point, it has a large s value and it is possible to be selected as an edge point. However, since the large distance between A and B, a huge gap between *a* and *a''* exists and could be used to filter the pseudo edge point.

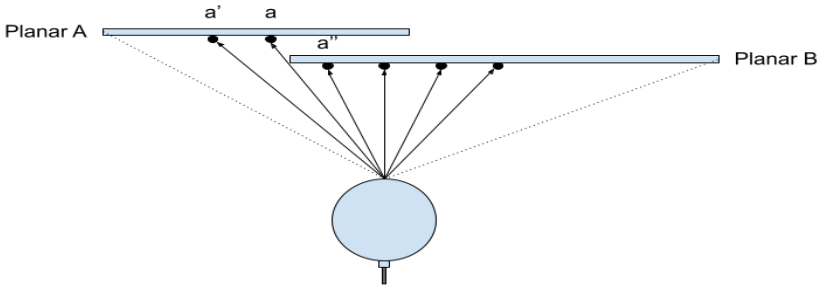

Figure 3: Point a on Planar A is actually a planar point. However, it seems to be on the edge of B from the view of the sensor so it is possible to be wrongly detected as an edge point.

4.2 Correspondences Estimation

Given two consecutive point clouds PC_t and PC_{t+1}, the next step is to find the correspondences based on the extracted feature point sets (E_t, Pl_t) and (E_{t+1}, Pl_{t+1}). At time stamp t+1, PC_t is projected into the coordinate S_{t+1} and the resulting point cloud is denoted as $PC_t{}'$ with the projected feature point sets denoted as $(E_t{}', Pl_t{}')$. PC_t', E_t' and $Pl_t{}'$ are now in the same coordinates with PC_{t+1}, $E_{t+1}$and Pl_{t+1}.

Given $e_{(i,k)}^{t+1}$ be a point of r_k^{t+1} from E_{t+1}, let $e_{(j,k)}^t{}'$ be the closet point of $e_{(i,k)}^{t+1}$ in $r_k^{t}{}'$ from $E_t{}'$ and $e_{(h,k\pm1)}^t{}'$ be the closet point of $e_{(i,k)}^{t+1}$ from the adjacent rings $r_{k\pm1}^t{}'$ of $r_k^{t}{}'$. Then $(e_{(j,k)}^t{}', e_{(h,k\pm1)}^t{}')$ forms the correspondence of $e_{(i,k)}^{t+1}$ in $E_t{}'$. The distance between the point and its corresponding edge is calculated as:

$$d_E = \frac{\left|\left(e_{(i,k)}^{t+1} - e_{(j,k)}^t{}'\right) \times \left(e_{(i,k)}^{t+1} - e_{(h,k\pm1)}^t{}'\right)\right|}{\left|e_{(j,k)}^t{}' - e_{(h,k\pm1)}^t{}'\right|}, \quad (2)$$

where $e_{(i,k)}^{t+1}$, $e_{(j,k)}^t{}'$ and $e_{(h,k\pm1)}^t{}'$ are the 3D coordinates of the corresponding points.
As for the planer correspondence, given $pl_{(i,k)}^{t+1}$ be a point of r_k^{t+1} from Pl_{t+1}, let $pl_{(j,k)}^t{}'$ and $pl_{(h,k)}^t{}'$ be the two closet points of $pl_{(i,k)}^{t+1}$ in $Pl_t{}'$, $pl_{(g,k\pm1)}^t{}'$ be the closet point of $pl_{(i,k)}^{t+1}$ from the consecutive rings $r_{k+1}^t{}'$ or $r_{k-1}^t{}'$. Then $(pl_{(j,k)}^t{}', pl_{(h,k)}^t{}', pl_{(g,k\pm1)}^t{}')$ forms the planar correspondence of $pl_{(i,k)}^{t+1}$. The distance from the point to its corresponding planar is:

$$d_{Pl} = \frac{\left|(pl_{(i,k)}^{t+1} - pl_{(j,k)}^t{}') \cdot ((pl_{(j,k)}^t{}' - pl_{(h,k)}^t{}') \times (pl_{(j,k)}^t{}' - pl_{(g,k\pm1)}^t{}')\right|}{\left|(pl_{(j,k)}^t{}' - pl_{(h,k)}^t{}') \times (pl_{(j,k)}^t{}' - pl_{(g,k\pm1)}^t{}')\right|}. \quad (3)$$

Then d_E and d_{Pl} are the optimization target of the algorithm.

4.3 Multi-map Transformation

The key problem is to estimate the optimal transformation matrices $\{T_1^0, T_2^0, \dots, T_n^0\}$ between each frame and the original frame. Since $T_{\mathrm{i}}^0 = \prod_{j=0}^{i} T_{\mathrm{i-j}}^{i-j-1}$, the problem is turned to estimate $\{T_1^0, T_2^1, \dots, T_n^{n-1}\}$, where $T := [\vec{t}, \vec{\theta}]^T$. $\vec{t}$ denotes the translation and $\vec{\theta}$ denotes the rotation angle around the x, y and z axes. Then given a pair of points $p_{\mathrm{i}}^t \in E_t \cup Pl_t$ and $p_{\mathrm{i}}^{t+1} \in E_{t+1} \cup Pl_{t+1}$, the transformation equation is:

$$p_{\mathrm{i}}^{t+1} = \mathbf{R} \cdot p_{\mathrm{i}}^t + T_{\mathrm{t}}^{t+1}[0:3], \tag{4}$$

And the rotation matrix $\boldsymbol{R}$ is calculated based on (Murray, Li et al. 2017):

$$\mathbf{R} = \mathbf{I} + \hat{\omega} \cdot sin\Theta + \hat{\omega}^2 \cdot (1 - cos\Theta), \tag{5}$$

where $\Theta = \| T_{\mathrm{t}}^{t+1}[4:6] \|$ and $\hat{\omega} = -\frac{T_{\mathrm{t}}^{t+1}[4:6]}{\|T_{\mathrm{t}}^{t+1}[4:6]\|}^T$ denotes the skew symmetric of rotation direction. According to equation Eq. 2 and Eq. 3, the optimization targets are d_E and d_{Pl}, the non-linear functions to relate the feature points and its correspondences could be formed as:

$$\begin{bmatrix} f_E(p_{\mathrm{i}}^{t+1}, T_{\mathrm{t}}^{t+1}) \\ f_{Pl}(p_{\mathrm{j}}^{t+1}, T_{\mathrm{t}}^{t+1}) \end{bmatrix} = \begin{bmatrix} d_E \\ d_{Pl} \end{bmatrix}, \tag{6}$$

where $i, j \in E^{t+1}, Pl^{t+1}$. Let $\boldsymbol{f} = \begin{bmatrix} f_E(p_{\mathrm{i}}^{t+1}, T_{\mathrm{t}}^{t+1}) & f_{Pl}(p_{\mathrm{j}}^{t+1}, T_{\mathrm{t}}^{t+1}) \end{bmatrix}^{\boldsymbol{T}}$, then the Jacobian matrix $\boldsymbol{J}$ of $\boldsymbol{f}$ could be derived based on T_{t}^{t+1} as $\boldsymbol{J} = \partial \boldsymbol{f} / \partial T_{\mathrm{t}}^{t+1}$ and the final optimization iterational problem is:

$$T_{\mathrm{t}}^{t+1} \leftarrow T_{\mathrm{t}}^{t+1} - \left(\boldsymbol{J}^{\boldsymbol{T}} \cdot \boldsymbol{J} + \lambda \cdot \mathrm{diag}(\boldsymbol{J}^{\boldsymbol{T}} \cdot \boldsymbol{J})\right)^{-1} \cdot \boldsymbol{J}^{\boldsymbol{T}} \cdot \begin{bmatrix} d_E \\ d_{Pl} \end{bmatrix}, \tag{7}$$

where λ is a pre-defined weight.

Once the transformation matrices $\{T_1^0, T_2^1, \dots, T_n^{n-1}\}$ is estimated by Eq. 7, we could get $\{T_1^0, T_2^0, \dots, T_n^0\}$ to reproject frame 1 to frame n back to frame 0. In section 3.2, we discussed that V16 only has 16 rings to cover a vertical angle range from -15° to 15°. Thus, there will be huge gaps between vertical rings, which is the main reason for data sparsity. To tackle the problem and build a dense indoor reconstruction map, we slightly rotate the LiDAR sensor around the x and y axes by β in both clockwise and counter-clockwise directions, as shown in Figure 4. This is done by controlling the locomotion of the quadrupedal robot in a corresponding way, see Figure 4. The scanning results from different views with different β values are projected back to the initial frame with $\beta = 0$. Let β_t denotes the horizontally rotated angle at timestamp t, then the final reconstructed map is $PC_{all} = [T_1^0 \cdot PC_1, T_2^0 \cdot PC_2, \dots, T_t^0 \cdot PC_t, \dots, T_n^0 \cdot PC_n]$. The final reconstructed map PC_{all} will be down sampled and sent to Unity through ROS sharp.

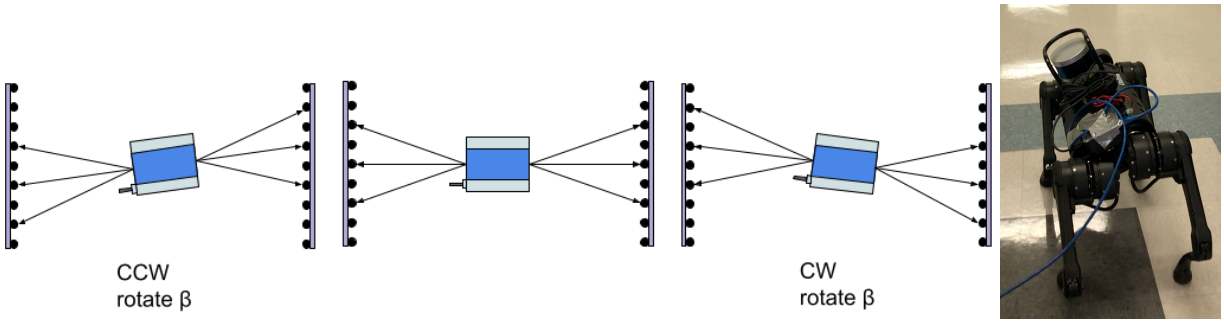

Figure 4: Rotation of the sensor in both CCW and CW direction enable the rings to cover the gaps of rings from the initial frame. Therefore, the vertical resolution could be largely improved.

4.4 LiDAR and HoloLens Model Alignment

After the LiDAR point cloud model is successfully acquired, it should be post-processed to consolidate into a single model with the AR's environmental mesh model, i.e., model registration. LiDAR maps acquired without sensor rotation are not adopted because of the lack of point clusters and extractable features. First,

the point clouds are smoothed with filters to remove excessive noise and outliers. For this process, the "Density-based spatial clustering of applications with noise" (DBSCAN) algorithm is first applied (Ester, Kriegel et al. 1996). This method aims to remove clusters with less content and far away from major clusters, especially for noises and artifacts generated from sensors interacting with unsatisfying objects. The second step is to extract feature points from each point cloud to reduce complexity and increase the effectiveness of the registration process. For this process, the feature descriptor named Fast Point Feature Histograms (FPFH) is used (Rusu, Blodow et al. 2008, Rusu, Blodow et al. 2009). Since it is a non-learning-based workflow based on feature histograms of values, it can be easily implemented to do real-time applications as an efficient way of qualifying geometric properties. Furthermore, FPFH can extract local pose-invariant features of each point that ensures the robustness of registration among a large scale of initial translation between two-point cloud maps.

For the last process, another non-learning-based method Fast Global Registration is applied to register two point clouds based on the feature points generated in the previous section (Zhou, Park et al. 2016). The point cloud model from LiDAR and AR's environmental models are then registered to form a consolidated and collaborative model. To be noted, the registration process also includes aligning the world coordinates of the HoloLens 2 HMD and that of the consolidated point cloud model. HoloLens 2 is equipped with a depth camera system and builds its 3D point cloud during the scanning and environmental understanding phase. The SDK, however, does not allow direct access to the raw point cloud model established by HoloLens. A workaround solution we found was a reverse engineering approach. First, we utilized HoloLens' MRTK's Spatial Awareness system to register the observer. This process allows us to access the 3D meshing model established by HoloLens. Then we apply an open-source mesh-to-point tool, point cloud library (PCL) (Durrant-Whyte and Bailey 2006), to convert the mesh model back into a dense point cloud model. Finally, the same workflow of aligning and merging LiDAR and depth-camera point clouds is repeated to further align the HoloLens-generated point cloud data with the scanned point cloud data. Once the coordinate system is reconciled, the consolidated point cloud model (from robot carried LiDAR and depth camera) is transformed (T) and rotated (R). The alignment errors will be corrected dynamically until a stopping threshold is achieved.

4.5 Data Visualization in AR

The last step of the proposed system involves rendering the results via an AR head-mounted display (HMD) device. The advantage of optical see-through AR HMD is to help the user visualize a virtual model on top of the real world, which provides great advantages for users to locate both themselves and virtual objects based on the real-world scene (Kim, Kang et al. 2017). To realize this design, two computers and one HoloLens 2 HMD are used. First, the point cloud data published from ROS containing transformation and color data of each spatial point are transferred to a Windows PC through ROS bridge and received by ROS# plug-in with-in Unity. This confronts the first challenge, which is streaming large point cloud data in real-time. The point cloud data from LiDAR and RGB-D sensors could be extensive when combined due to the limitation of mobile processors and web sockets. An ethernet cable is used to bridge the data stream between the Linux machine and the windows machine. The second step is to process the raw point cloud data into the correct form that could be rendered with the Unity game engine. The rendering of point cloud is intrinsically difficult due to its scattering nature and takes up a lot of RAM and computing power if not handled correctly (Griffiths and Boehm 2019). We incorporated the VisualEffect (VFX) function in Unity to both register the transformation of each point and render them individually. The advantage of VFX is for rendering large-scale point clouds (up to millions of points) with the help of shader graph in comparison to the conventional particle system. In our previous study, we found that the particle system could only handle particles on the scale of thousands, and the frame rate drops to an unbearable state (5 Hz). The High-Definition Render Pipeline (HDRP) is also required for VFX to work. In the Unity engine, a game object with updating transformation from the SLAM's trajectory ensures the stability of point cloud model, and the points are registered and rendered accordingly. The third step is to stream the real-time rendered video data to a HoloLens 2 device through Wi-Fi connection. The incorporation of HoloLens 2 with Unity is developed with the help of Mixed Reality Toolkit (MRTK). The resultant visualization could be described in this way: a digital twin model of the target space in form of a point cloud model registered on top of the real world and could be seen from a distance regardless of occlusion and motion of the user. This enables a seeing-through wall sensation for the operator (Pu, Wei et al. 2021). In terms of the real-time capability, the proposed system

could handle point cloud data streaming with short delay (5 seconds) and have continuous streaming capability. Raw sensor data were also visualized with updating point cloud (> 1 Hz frequency) to ensure the situational awareness of surrounding environments (from 360° LiDAR) and frontal situation (from depth camera). Furthermore, the visualization cloud is shared with multiple agents collaboratively or hierarchically for better team task performance.

5. RESULTS AND DISCUSSION

In this session, we show the qualitative results of our reconstruction method and reality capture speed in Unity. In Figure 5, we present our reconstruction map. The detailed spatial information was captured including the shape of objects. The fused map included the scanning point cloud of approximately 50 frames. The merged point cloud was down sampled by cubic voxels with the size of 0.2m*0.2m*0.2m. The region covered by a voxel was replaced by its center point if it contained at least 5 points. Otherwise, the voxel would be dropped, leaving the region empty. The point density of rings from either raw scanning or reconstruction were insufficient to represent the vertical spatial changes. As a result, the reconstruction map using our proposed fusion and data augmentation method was applied. It was able to capture the detailed vertical spatial changes by adding new rings of scanning results from different views. Also, the floor and ceiling structure could be captured by changing the viewing angle when gathering the frames for reconstruction. Finally, processes discussed in sections 4.4 and 4.5 were applied to render the 3D models in HoloLens 2.

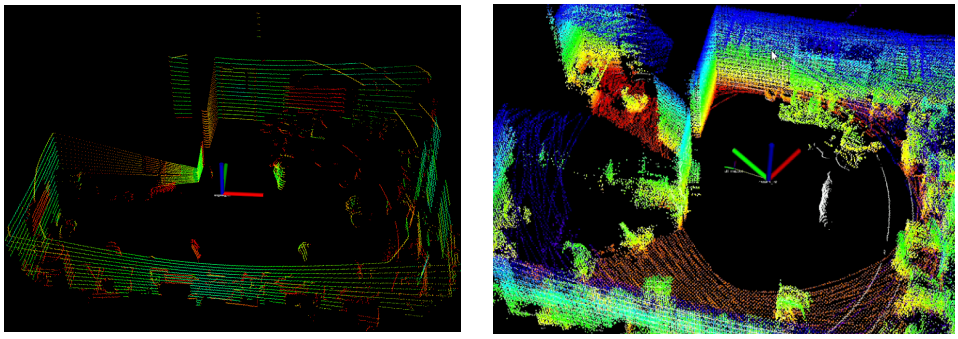

Figure 5: The comparison of raw scanning result (on the left) and the reconstructed map (on the right).

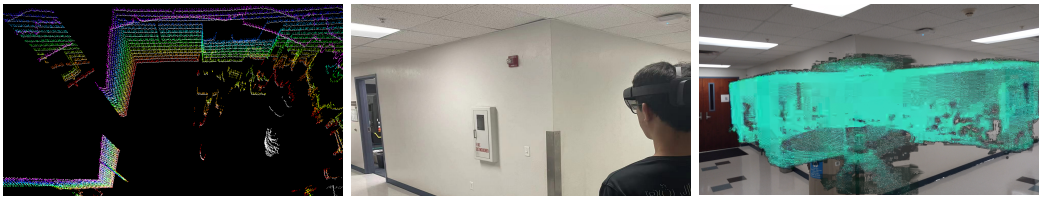

Figure 6: The real-time scanning result on the constructed indoor map in ROS and HoloLens.

Figure 6 shows the real-time scanning result based on the rebuilt map. The left one visualizes the reconstructed map in ROS. The color points form the 3D map, and the white dots are the dynamic scanning result. The middle one is the captured reality map in Unity with the blue points set to be the dynamical scanning results and the left one is the visualization of the rebuilt map in HoloLens. The transmission rate is 1Hz which is a near real-time speed with an average of 51,626 points for the map and 27,791 points for the dynamical scan. Our proposed method could largely increase the density of 3D indoor scanning results with LiDAR at a near real-time speed. The reprojection of frames with different viewing angles could effectively fill the spare space between the rings of raw LiDAR scanning point cloud and provide rich information for capturing 3D spatial information. A demo video can be found at: https://youtu.be/rjuw9Hi5xrU. It shows that how a high-resolution point cloud model was built with only a low-resolution LiDAR, how the robot platform was used to carry the mobile LiDAR, and how the result was rendered in HoloLens 2 with high fidelity.

6. ACKNOWLEDGMENTS

This material is supported by National Institute of Standards and Technology (NIST) under grant 70NANB21H045. Any opinions, findings, conclusions, or recommendations expressed in this article are those of the authors and do not reflect the views of NIST.

7. REFERENCE

Armeni, Iro, Sasha Sax, Amir R Zamir, and Silvio Savarese. "Joint 2d-3d-Semantic Data for Indoor Scene Understanding." *arXiv preprint arXiv:1702.01105* (2017).

Bekele, Mafkereseb Kassahun, and Erik Champion. "A Comparison of Immersive Realities and Interaction Methods: Cultural Learning in Virtual Heritage." [In English]. Review. *Frontiers in Robotics and AI* 6 (2019-September-24 2019). https://doi.org/10.3389/frobt.2019.00091. https://www.frontiersin.org/article/10.3389/frobt.2019.00091.

Bolourian, Neshat, and Amin Hammad. "Lidar-Equipped Uav Path Planning Considering Potential Locations of Defects for Bridge Inspection." *Automation in Construction* 117 (2020). https://doi.org/10.1016/j.autcon.2020.103250.

Caesar, Holger, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan*, et al.* "Nuscenes: A Multimodal Dataset for Autonomous Driving." Paper presented at the Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2020.

Dai, Angela, Christian Diller, and Matthias Nießner. "Sg-Nn: Sparse Generative Neural Networks for Self-Supervised Scene Completion of Rgb-D Scans." Paper presented at the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020.

Durrant-Whyte, Hugh, and Tim Bailey. "Simultaneous Localization and Mapping: Part I." *IEEE robotics & automation magazine* 13, no. 2 (2006): 99-110.

Ester, Martin, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise." Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, Portland, Oregon, AAAI Press, 1996.

Geiger, Andreas, Philip Lenz, and Raquel Urtasun. "Are We Ready for Autonomous Driving? The Kitti Vision Benchmark Suite." Paper presented at the 2012 IEEE conference on computer vision and pattern recognition, 2012.

Griffiths, David, and Jan Boehm. "Synthcity: A Large Scale Synthetic Point Cloud." *arXiv preprint arXiv:1907.04758* (2019).

Hua, Minjie, Yibing Nan, and Shiguo Lian. "Small Obstacle Avoidance Based on Rgb-D Semantic Segmentation." Paper presented at the Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops, 2019.

Huang, Kemiao, and Qi Hao. "Joint Multi-Object Detection and Tracking with Camera-Lidar Fusion for Autonomous Driving." Paper presented at the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2021.

Jiang, Yao, Wenbo Zhang, Keren Fu, and Qijun Zhao. "Meanet: Multi-Modal Edge-Aware Network for Light Field Salient Object Detection." *Neurocomputing* 491 (2022): 78-90. https://doi.org/10.1016/j.neucom.2022.03.056.

Jonasson, Emil T., Luis Ramos Pinto, and Alberto Vale. "Comparison of Three Key Remote Sensing Technologies for Mobile Robot Localization in Nuclear Facilities." *Fusion Engineering and Design* 172 (2021). https://doi.org/10.1016/j.fusengdes.2021.112691.

Karkus, Peter, Shaojun Cai, and David Hsu. "Differentiable Slam-Net: Learning Particle Slam for Visual Navigation." Paper presented at the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021.

Kim, Kyungki, Sungjin Kim, and Daniel Shchur. "A Uas-Based Work Zone Safety Monitoring System by Integrating Internal Traffic Control Plan (Itcp) and Automated Object Detection in Game Engine Environment." *Automation in Construction* 128 (2021). https://doi.org/10.1016/j.autcon.2021.103736.

Kim, Soo Kyun, Shin-Jin Kang, Yoo-Joo Choi, Min-Hyung Choi, and Min Hong. "Augmented-Reality Survey: From Concept to Application." *KSII Transactions on Internet and Information Systems (TIIS)* 11, no. 2 (2017): 982-1004.

Murray, Richard M, Zexiang Li, and S Shankar Sastry. *A Mathematical Introduction to Robotic Manipulation.* CRC press, 2017.

Pu, George, Paul Wei, Amanda Aribe, James Boultinghouse, Nhi Dinh, Fang Xu, and Jing Du. "Seeing through Walls: Real-Time Digital Twin Modeling of Indoor Spaces." Paper presented at the 2021 Winter Simulation Conference (WSC), 2021.

Rodrigues, André Montes, Mário Nagamura, Luis Gustavo Freire Da Costa, Rodrigo Rodrigues Gesuatto Faria, Pier Luigi Nakai Ricchetti, Eric Nozomi Tatsuta, Roseli De Deus Lopes, and Marcelo Knorich Zuffo. "Batmen X the Puzzler-Escaping Ar's Drawbacks with Augmented Virtuality and Low Cost Sensors." Paper presented at the 2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR), 2019.

Rusu, Radu Bogdan, Nico Blodow, and Michael Beetz. "Fast Point Feature Histograms (Fpfh) for 3d Registration." Paper presented at the 2009 IEEE international conference on robotics and automation, 2009.

Rusu, Radu Bogdan, Nico Blodow, Zoltan Csaba Marton, and Michael Beetz. "Aligning Point Cloud Views Using Persistent Feature Histograms." Paper presented at the 2008 IEEE/RSJ international conference on intelligent robots and systems, 2008.

Sajjan, Shreeyak, Matthew Moore, Mike Pan, Ganesh Nagaraja, Johnny Lee, Andy Zeng, and Shuran Song. "Clear Grasp: 3d Shape Estimation of Transparent Objects for Manipulation." Paper presented at the 2020 IEEE International Conference on Robotics and Automation (ICRA), 2020.

Schiavi, Barbara, Vincent Havard, Karim Beddiar, and David Baudry. "Bim Data Flow Architecture with Ar/Vr Technologies: Use Cases in Architecture, Engineering and Construction." *Automation in Construction* 134 (2022). https://doi.org/10.1016/j.autcon.2021.104054.

Shi, Shaoshuai, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. "Pv-Rcnn: Point-Voxel Feature Set Abstraction for 3d Object Detection." Paper presented at the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020.

Song, Shuran, Samuel P Lichtenberg, and Jianxiong Xiao. "Sun Rgb-D: A Rgb-D Scene Understanding Benchmark Suite." Paper presented at the Proceedings of the IEEE conference on computer vision and pattern recognition, 2015.

Sun, Pei, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo*, et al.* "Scalability in Perception for Autonomous Driving: Waymo Open Dataset." Paper presented at the Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2020.

Tan, Weikai, Nannan Qin, Lingfei Ma, Ying Li, Jing Du, Guorong Cai, Ke Yang, and Jonathan Li. "Toronto-3d: A Large-Scale Mobile Lidar Dataset for Semantic Segmentation of Urban Roadways." *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (2020): 797-806.

Vora, Sourabh, Alex H Lang, Bassam Helou, and Oscar Beijbom. "Pointpainting: Sequential Fusion for 3d Object Detection." Paper presented at the Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2020.

Xiao, Cheng, and Zhang Lifeng. "Implementation of Mobile Augmented Reality Based on Vuforia and Rawajali." Paper presented at the 2014 IEEE 5th International Conference on Software Engineering and Service Science, 2014.

Yang, Ming-Der, Chih-Fan Chao, Kai-Siang Huang, Liang-You Lu, and Yi-Ping Chen. "Image-Based 3d Scene Reconstruction and Exploration in Augmented Reality." *Automation in Construction* 33 (2013): 48-60. https://doi.org/10.1016/j.autcon.2012.09.017.

Zhang, Ji, and Sanjiv Singh. "Loam: Lidar Odometry and Mapping in Real-Time." Paper presented at the Robotics: Science and Systems, 2014.

Zhou, Qian-Yi, Jaesik Park, and Vladlen Koltun. "Fast Global Registration." Paper presented at the European conference on computer vision, 2016.

Zhou, Yin, and Oncel Tuzel. "Voxelnet: End-to-End Learning for Point Cloud Based 3d Object Detection." Paper presented at the Proceedings of the IEEE conference on computer vision and pattern recognition, 2018.

Zhu, Jingwei, Joachim Gehrung, Rong Huang, Björn Borgmann, Zhenghao Sun, Ludwig Hoegner, Marcus Hebel, Yusheng Xu, and Uwe Stilla. "Tum-Mls-2016: An Annotated Mobile Lidar Dataset of the Tum City Campus for Semantic Point Cloud Interpretation in Urban Areas." *Remote Sensing* 12, no. 11 (2020). https://doi.org/10.3390/rs12111875.