Real-time Monitoring of Point Cloud Registration in Mixed Reality

Hanbeom Chang[1,2], Hansol Lim[1,2], and Jongseong Brad Choi[1,2,*]

[1] Mechanical Engineering, State University of New York, Stony Brook, Stony Brook, USA
{hanbeom.chang, hansol.lim, jongseong.choi}@stonybrook.edu

[2] Mechanical Engineering, State University of New York, Korea, Incheon, South Korea
jongseong.choi@sunykorea.ac.kr

*Corresponding Author

Abstract. Advances in 3D sensing technologies, such as LiDAR and RGB-D cameras, alongside increased computational power, have significantly improved Simultaneous Localization and Mapping (SLAM) algorithms, enabling 3D environment reconstruction. Indoors, however, implementing Visual-LiDAR SLAM algorithms poses challenges, including errors from rapid scene changes and blind spots due to the parallax effect. Mixed reality (MR) devices can be a breakthrough for such issues providing users the ability to interact with virtual objects within real-world settings, so-called holograms. In this study, we developed a method to achieve instantaneous visualization of point cloud registration reflecting on MR devices aiming for a thorough indoor 3D reconstruction. This paper analyzes parameter values and visualized data quantities to optimize real-time performance and visualization quality. With this, point cloud registration can be viewed in the real world and real-time, while data is being collected enabling users to react to any error.

Keywords: Navigation and SLAM, Human-robot interaction, Sensors Technology

1 Introduction

Advancements in 3D sensing technologies such as LiDAR and RGB-D cameras have made capturing and processing 3D data more accessible and efficient [1-5]. Simultaneously, improved computational power has advanced Simultaneous Localization and Mapping (SLAM) algorithms, enabling continuous tracking of sensor positions and the reconstruction of dense 3D point cloud maps representing real-world objects or structures. Widely used SLAM algorithms—such as Visual-SLAM [6-7], LiDAR-SLAM [8], and Visual-LiDAR SLAM [9]—are employed for this purpose using either vision or LiDAR sensors. Recently, mixed reality (MR) devices, such as the Microsoft HoloLens 2 and Apple Vision Pro, have initiated a new era in 3D visualization, allowing users to blend virtual objects with the real world,

creating interactive experiences [10, 11]. In this study, we aim to visualize and monitor the 3D reconstruction of indoor environments, specifically using Visual-LiDAR SLAM algorithms, on an MR device in real-time. However, this task presents challenges, especially regarding the visualization of complex 3D models and simultaneous localization on resource-limited MR devices.

While SLAM algorithms are highly robust and accurate for outdoor environments, they face several limitations in indoor settings. Indoor environments typically provide less data variety, with structures closer to the LiDAR and camera sensors, resulting in rapid scene changes and reduced overlap between scenes. Vision sensors, due to their limited field of view, often introduce issues such as blind spots, double walls, and drift in the point cloud map. Additionally, both LiDAR and camera sensors generate large datasets to enhance the quality of point clouds, creating data transmission challenges. High data volume, along with the need to synchronize depth and color data, can lead to latency or imprecision in real-time applications. Bandwidth limitations and communication bottlenecks between sensors and the MR device further affect response time and the overall quality of the visualization.

This study proposes an opportunity to harness MR devices to monitor the real-time 3D reconstruction process using a Visual-LiDAR SLAM algorithm. The unique ability of MR devices to localize virtual objects—such as point clouds—within the real-world coordinate system, coupled with their capability to ensure that virtual objects interact realistically with physical structures, offers a valuable advantage. While existing visualizing systems on traditional monitors can display point clouds, they often fail to localize these clouds within the real world, causing confusion in scale and appearance. In contrast, MR devices can display colorized point clouds at their precise locations in the environment, facilitating a more intuitive interaction with the 3D reconstruction process.

Achieving real-time 3D reconstruction and visualization on MR devices presents several challenges. The limited computing power and graphical capabilities of MR hardware make it difficult to render large, high-resolution point clouds without compromising real-time performance. Additionally, ensuring the accurate localization of the point cloud in the real world, while optimizing data transmission to minimize latency, remains a complex problem. Balancing performance and quality are critical to enabling efficient and effective visualization in indoor environments.

The goal of this work is to develop a system capable of real-time 3D reconstruction and visualization of colorized point clouds on MR devices. The system will employ advanced data processing techniques to ensure efficient transmission with TCP-Endpoint, 3D reconstruction, and accurate rendering of point clouds at the correct location by Oriented Fast and Rotated BRIEF (ORB) algorithm [12], while maintaining high fidelity in color information. By addressing computational and bandwidth constraints, the system aims to provide smooth, real-time monitoring of indoor environments using MR devices.

The proposed system offers several key advantages. First, it enables users to interact with indoor 3D reconstructions in a more immersive and spatially accurate way, enhancing spatial understanding and decision-making. Second, the system

allows for early detection of common errors in indoor SLAM algorithms, reducing the need for revisits or rescans of the environment, saving time and effort. Finally, the identification of optimized parameters—such as the number of points or the quality of the colorized point cloud—will serve as a foundation for future research in this domain.

This paper contributes to the growing field of mixed reality by presenting a system for real-time 3D reconstruction monitoring in indoor environments. The development of a low-latency pipeline ensures efficient point cloud transmission to MR devices. Additionally, the implementation of optimized parameters for colorizing point clouds improves visual fidelity. The system's demonstrated ability to operate in real-time on MR hardware strikes a balance between performance and quality. Through these contributions, the study advances real-time 3D visualization techniques for mixed reality platforms.

The paper follows: a System Overview of the system with detailed steps, Experimental Validation, Conclusion, and Reference.

2 System Overview

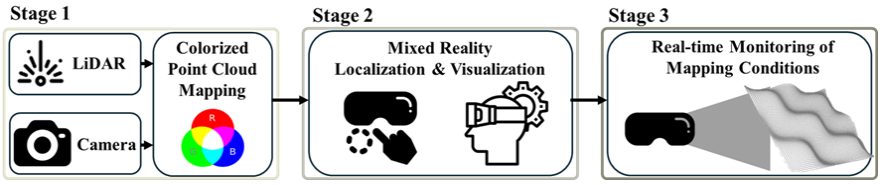

Fig. 1. Overview for visualizing mapping conditions in mixed reality device

This section provides an overview of MpcMR (Monitoring point cloud in Mixed Reality) system in real-time. The system overview is presented in Figure 1. The framework is in three stages, colorizing point cloud, localization, and visualization of colorized point cloud on MR device, and real-time monitoring of the mapping conditions. To generate a precise and realistic colorized point cloud map, LiDAR and RGB camera must be tightly coupled. With proper calibration, the mapped data is sent to Unity app through TCP-Endpoint. Simultaneously, the RGB camera and MR device's camera applies ORB feature matching and detection and estimate their pose in the global map. This localization process will ensure the proper location of colorized point clouds in the real world. After the colorizing point cloud and localization, the user with the scanner and MR device headset scan around the indoor sites to scan, generate point cloud, and monitor the 3D reconstruction process in real-time.

2.1 Point Cloud Colorization

The first stage contains extrinsic calibration between the LiDAR and RGB camera to create a custom scanner and colored point cloud generation. LiDAR's point cloud contains depth information such as x, y, and z coordinates. The RGB camera provides visual information like images or videos pixel wise. To combine the two different data, the extrinsic relationship between the sensors is required. Before extrinsic calibration, the intrinsic parameters of camera are needed. This includes focal lengths (f_x, f_y), principal points (c_x, c_y), and distortion coefficients. The Robot Operating System package has a calibration package [13] for camera intrinsic parameter. This package applies OpenCV camera calibration to calculate the intrinsic parameters by capturing few images.

The extrinsic parameters of LiDAR and RGB camera contains 3x3 rotation matrix and translation vector as described in equation (1):

$$s\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{1}$$

where x, y is the 2D image coordinate from the camera image and X, Y, and Z indicates the 3D world coordinate from the LiDAR point cloud. However, sensor fusion is not an easy task to do. We have applied direct_visual_lidar_calibration [14], a ROS package for extrinsic calibration. The package replays the sensor data to create a point cloud map and match the image using OpenCV, GTSAM, Ceres Solver, and Iridescence. Since both LiDAR and RGB camera are fixed on the sensor mount the rotation matrix and translation vector does not change.

Next step is to receive the point cloud and image from the sensors. For the convenience of data management, ROS drivers of LiDAR and RGB camera are used. The drivers publish sensor_msgs/PointCloud2, sensor_msgs/Image, and sensor_msgs/CameraInfo messages from the LiDAR and RGB camera. These ROS topics contain raw sensor data which will be used for generating colorized point clouds. R^{3}LIVE is a robust and reliable color point cloud mapping tool by easily subscribing to the LiDAR and camera ROS topics. This handles sensors' frequencies, color mapping calculations, and outliers to reduce the error and provide a robust 3D colorized point cloud map.

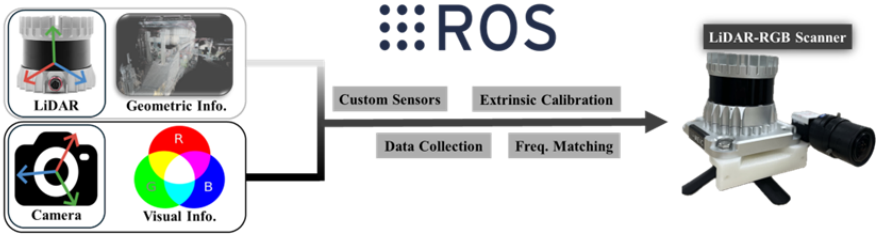

Fig. 2. LiDAR-RGB Scanner for Colorizing Point Cloud and Mapping

2.2 Mixed Reality Localization and Visualization

In this step, localization of the colorized point cloud from the previous step is a key point for visualization. Localization is done in a step of feature detection & matching, pose estimation, and filtering for stability. For feature detection and matching, we use ORB algorithm and Brute-Force (BF) matcher for securing the real-time localization. Moreover, pose estimation is done with triangulation method, to recognize three positions of feature, RGB camera, and MR device. Filtering used are Kalman filter and Median filter to smooth the result and reduce outliers.

Detecting features and matching them between the two camera frames has multiple methods such as SIFT [15], SURF [16], or ORB. ORB is built based on SIFT algorithm and composed of FAST key point detection and BRIEF descriptor. ORB apply FAST, the key point detection using intensity difference as:

$$|I(p) - I(x_i)| > T \; for \; x_i \in \{x_1, x_2, \dots, x_{16}\} \tag{2}$$

where $I(p)$ is the intensity of pixel p, $I(x_i)$ are the intensities of the pixels in the 16-pixel ring around p, and T is the threshold for intensity difference.

BRIEF descriptor generates binary descriptors for each key point. For two pixels p_i and p_j around a key point, the descriptor for each pair is calculated as:

$$f(p_i, p_j) = \begin{cases} 1 & if \; I(p_i) < I(p_j) \\ 0 & otherwise \end{cases} \tag{3}$$

Brute Force Matching (BF) uses a distance metric to compare the binary descriptors generated by ORB and finds the best match between key points in the two images. The Hamming distance is used:

$$Hamming(D_1, D_2) = \sum_{i=1}^{n} (D_1[i] \oplus D_2[i]) \tag{4}$$

where D_1 and D_2 are the two binary descriptors, $D_1[i]$ and $D_2[i]$ are the bits of the descriptors, n is the length of the binary descriptor, and $\oplus$ represents the XOR operation.

For pose estimation, Triangulation is used to estimate the 3D points of the matched key points from the two camera views. The given camera projection matrices P_1 and P_2, and the corresponding 2D points x_1 and x_2 in the two images, we can triangulate the 3D point X using:

$$\lambda_1 x_1 = P_1 X \; and \; \lambda_2 x_2 = P_2 X \tag{5}$$

where λ_1 and λ_2 are depths for the points in the respective camera images. Since camera projection matrices are given in the intrinsic camera parameters, we can track the pose of their positions.

However, to properly apply the point cloud on MR device, finding the relative rotation and translation bssetween the two cameras is required. Perspective-n-Point (PnP) is used to relate the 3D points X_1, X_2,…, X_n in the world and 2D image points x_1, x_2,…, x_n:

$$x_i = K(RX_i + t) \; and \; Re - projection \; Error = \sum_{i=1}^{n} \|x_i - \widehat{x_i}\|^2 \tag{6}$$

where K is the camera intrinsic matrix, R is the rotation matrix, t is the translation vector, and $\widehat{x_i}$ are the projected 3D points using the estimated pose.

After estimating the pose of the cameras, Kalman filter and Median filter are used for the stability of the point cloud. The Kalman filter smooths the pose estimates by predicting the next state and updating it based on new measurements to reduce the noise. Median filter stabilizes the pose by taking the median of the last ten frames of rotation and translation to reject outliers and prevent sudden jumps.

For each component of rotation matrix R and translation vector t, the filter computes:

$$x_{filtered} = median\left(x_{i-n//2}, \dots, x_{i+n//2}\right) \tag{7}$$

where $x_{i-n//2}, \dots, x_{i+n//2}$ are the values in a window around the current data point x_i, and n is the size of the window and its set as 10 frames.

2.3 Real-time Monitoring of Mapping Conditions

Within the two steps, ROS machine colors point cloud with LiDAR and RGB camera and calculates localization to estimate the pose of the scanner and MR device. For the real-time monitoring of this data, MR device's subscription to the ROS data is needed. ROS-TCP-Endpoint [17] and ROS-TCP-Connector [18] is an option to send and receive the data in a ROS message type and visualization in Unity app. These packages use TCP-Endpoint connection between the ROS machine and Unity (or MR device). The package subscribes colorized point cloud messages such as sensor_msgs/PointCloud2 calculated from ROS and publishes sensor_msgs/Image to ROS for the localization. To ensure the performance of a real-time monitoring system, we must consider the quality of the point cloud and latency of the data transfer. Subscription of colorized point cloud is limited to half of the original density to reduce the density in the Unity side.

Lastly, to evaluate the optimized parameters for ensuring real-time monitoring, a method for calculating the density of point cloud is required. Mean Nearest-Neighbor Distance (MNN Distance) is a metric used to quantify how tightly packed points are in a point cloud. The Euclidean distance between a point p_i and its nearest neighbor p_j, and for each point p_i, the mean nearest-neighbor distance is calculated:

$$D_{mean} = \frac{1}{N}\sum_{i=1}^{N} d_{avg}(p_i) \tag{8}$$

where N is the set of each point nearest neighbors, d_{avg} is average distance between p_i and its neighbors.

3 Experimental Validation

In this section, an experimental validation of generating a colorized point cloud and visualizing it on the MR device is provided. This includes experimental environment selection, equipment setup, and software configuration and the validation results.

3.1 Experimental Setup

For the experimental validation, an adjust size of an indoor environment with complicated objects such as desks, interior objects, and pillars are required. The MEIC laboratory in the State University of New York, Korea, Incheon (SUNY Korea) is an appropriate indoor environment to scan, localize, and visualize the point cloud. In the validation, only the main hall including the three sides of the walls will be used. As Figure 3 shows, the laboratory area is about 120 m^2, with desks, pillars, interior objects, robot arm, and other equipment.

Fig. 3 State University of New York, Korea, Incheon, MEIC Laboratory

3.2 Equipment Setup

ROS Machine (Remote Machine)		FLIR Blackfly S		Ouster OS0-32		Microsoft HoloLens 2	
PC	Intel NUC 11th	Resolution	1440 x 1080	Horizontal Scan	360°	Processor	Qualcomm Snapdragon 850
Processor	Intel 11th i7-1165G7 2.80GHz x8	Frame Rate	60 FPS	Vertical Scan	90° (±45°)	HPU	2nd Holographic Process Unit
Memory	16 GB	Field of View	70°	Max. Range	35 m	Resolution	2k 3:2
Disk	500 GB	Camera Sensor	SONY IMX273	Resolution	1024 x 20 Hz	Memory	4 GB LPDDR4x DRAM
OS	Ubuntu 20.04.6 LTS	CMOS Sensor Size	9 ½'	Data Transfer	655,360 points/s	Sensors	5 RGB Camera, 1 Depth, IMU

Fig. 4 System Specifications

As described in Figure 4, our system utilizes one ROS machine, *FLIR Blackfly S* for RGB camera, *Ouster OS0-32* for LiDAR and *Microsoft HoloLens 2* for MR device. The CPU and RAM memory of ROS machine manages the localization and generating colorized point cloud to be transferred to Unity (HoloLens 2 app). The LiDAR and RGB camera are connected to the ROS machine to send dense point clouds and colors to generate colorized point cloud. HoloLens 2 runs the Unity app that connects to ROS machine with TCP-Endpoint for monitoring colorized point cloud on HoloLens 2 glasses.

3.3 Point Cloud Visualization

The Unity app has 4 main Unity assets: *ROS-TCP-Connector*, *PointCloudVisualizer*, *HoloLens2Visualizer*, and *Mixed Reality Toolkit (MRTK)*. *PointCloudVisualizer* is an asset built with Unity C# script visualizes multiple point clouds based on *ROS-TCP-Endpoint* to receive the colorized point cloud and localization data in real-time. *HoloLens2Visualizer* is also a Unity C# script that is built based on HoloLens 2 Sensor Streaming (HL2SS) [19] to access low-level sensor data and stream

the HoloLens 2 camera raw data. The script will allow access to *HoloLens 2* sensors including cameras, IMU (Inertial Measurement Unit) and depth camera. Lastly, *MRTK* [20] is a Microsoft-driven project that provides a set of components and features, used to accelerate cross-platform MR app development in Unity. The assets are stored in Unity project and built as a .appx bundle app to be deployed in HoloLens 2. ROS plays sensor drivers including LiDAR and RGB camera, *HoloLens 2* topic publisher, a localization code, and TCP-Endpoint. *HoloLens 2* runs Unity app that connects to ROS with the messages from sensors and transfer data. The visualized result is shown in Fig. 5.

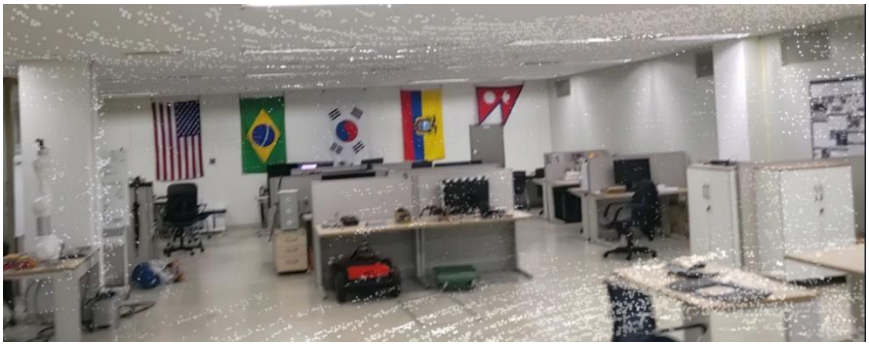

Fig. 5 MpcMR Visualization in HoloLens 2

3.4 Validation Results

The R^{3}LIVE color mapping package provides dense and realistic color of point clouds scanned from the LiDAR and RGB camera. However, the TCP-Endpoint depends on the Wi-Fi connection and the bandwidth of it. Therefore, we have made optimized parameter guidelines of the number of the point cloud, density, and delay time in *HoloLens 2* to ensure the real-time data visualization. The density of the point cloud can be calculated with mean nearest-neighbor distance from the ROS machine side, and the delay time is calculated from the initialization of point cloud visualization to the maximum number of point clouds visualized. As Table 1 shows, we have found out how parameters from ROS side and how many points are visualized on *HoloLens 2* change the latency of the point cloud visualization. Moreover, we have played back the sensor data to measure the latency and density.

Table 1 MpcMR Visualization Quality Difference of Parameters

Reduction of R3LIVE Params.	**0%**	**10%**	**30%**	**40%**	**50%**
# of Visualized Points	**10000**	**8000**	**6000**	**6000**	**5000**
Delay time (s)	10.01	9.20	8.93	6.23	5.94
Density of Point Cloud	0.0438	0.0434	0.0430	0.0426	0.0424
Number of points	82978	82689	82201	67221	54373

The reduced R^{3}LIVE parameters are increased minimum distance for every two points, decreased maximum points for tracking, and increased filter size of the LiDAR point clouds. The entire result shows that the reduced parameters of R^{3}LIVE and visualized points in *HoloLens 2* reduce the latency, but the density and entire number of point cloud is also reduced. For the proper amount of quality and delay time, the option is to have a 40% reduction in R^{3}LIVE parameters, and 6,000 visualized points on HoloLens 2.

4 Conclusion

In conclusion, we have successfully created a system to monitor the 3D reconstruction on MR device in real-time under conditions. The localization process through feature detection and matching, and multiple filters effectively stabilize the colorized point cloud to be set on the real objects. The limitations of computing power in *HoloLens 2* and the bandwidth of the internet connectivity cause delay in visualization and restriction of the number of the visualized point cloud. However, the visualized point cloud map on *HoloLens 2* with optimized parameters provides a thorough understanding in scanning process and checks for any errors during the scanning process. Thus, the MpcMR system has proved to effectively provide a denser and realistic colorized point cloud map afterwards and reduce the trip for rescanning due to the problem of double walls and drifting in SLAM.

References

1. Fiocco, G., & Smullin, L. D. (1963). Detection of scattering layers in the upper atmosphere (60–140 km) by optical Radar. Nature, 199(4900), 1275–1276. https://doi.org/10.1038/1991275a0

2. Russell, P. B., Swissler, T. J., & McCormick, M. P. (1979). Methodology for error analysis and simulation of Lidar Aerosol Measurements. Applied Optics, 18(22), 3783. https://doi.org/10.1364/ao.18.003783

3. Abellán, A., Calvet, J., Vilaplana, J. M., & Blanchard, J. (2010). Detection and spatial prediction of rockfalls by means of terrestrial laser scanner monitoring. Geomorphology, 119(3–4), 162–171. https://doi.org/10.1016/j.geomorph.2010.03.016

4. Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I., & Leonard, J. J. (2016). Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age. IEEE Transactions on Robotics, 32(6), 1309–1332. https://doi.org/10.1109/TRO.2016.2624754

5. Durrant-Whyte, H., Rye, D., & Nebot, E. (1996). Localization of autonomous Guided Vehicles. Robotics Research, 613–625. https://doi.org/10.1007/978-1-4471-0765-1_69

6. Lategahn, H., Geiger, A., & Kitt, B. (2011). Visual slam for autonomous ground vehicles. 2011 IEEE International Conference on Robotics and Automation. https://doi.org/10.1109/icra.2011.5979711

7. Mur-Artal, R., Montiel, J. M., & Tardos, J. D. (2015). Orb-Slam: A versatile and accurate monocular slam system. IEEE Transactions on Robotics, 31(5), 1147–1163. https://doi.org/10.1109/tro.2015.2463671

8. Hess, W., Kohler, D., Rapp, H., & Andor, D. (2016). Real-time loop closure in 2D LIDAR SLAM. 2016 IEEE International Conference on Robotics and Automation (ICRA). https://doi.org/10.1109/icra.2016.7487258
9. Lin, J., & Zhang, F. (2022). R3live: A robust, real-time, RGB-colored, LIDAR-inertial-visual tightly-coupled state estimation and mapping package. 2022 International Conference on Robotics and Automation (ICRA). https://doi.org/10.1109/icra46639.2022.9811935
10. Chen, Y., Li, X., Chen, L., Ma, Y., & Xiao, X. (2020). HoloLens-based System for Real-time Monitoring of Geological Drilling. Automation in Construction, 113, 103145. https://doi.org/10.1016/j.autcon.2020.103145
11. Mullen, L., Scheid, J., & Lee, S. (2021). Augmented Reality for Real-time LiDAR and Point Cloud Visualization. Journal of the Society for Information Display, 29(3), 211-220. https://doi.org/10.1002/jsid.1001
12. E. Rublee, V. Rabaud, K. Konolige and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," 2011 International Conference on Computer Vision, Barcelona, Spain, 2011, pp. 2564-2571, doi: 10.1109/ICCV.2011.6126544.
13. https://wiki.ros.org/camera_calibration
14. https://github.com/koide3/direct_visual_lidar_calibration
15. Lowe, D.G. Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision 60, 91–110 (2004). https://doi.org/10.1023/B:VISI.0000029664.99615.94
16. Bay, H., Tuytelaars, T., Van Gool, L. (2006). SURF: Speeded Up Robust Features. In: Leonardis, A., Bischof, H., Pinz, A. (eds) Computer Vision – ECCV 2006. ECCV 2006. Lecture Notes in Computer Science, vol 3951. Springer, Berlin, Heidelberg. DOI: https://doi.org/10.1007/11744023_3
17. https://github.com/Unity-Technologies/ROS-TCP-Endpoint
18. https://github.com/Unity-Technologies/ROS-TCP-Connector
19. https://github.com/jdibenes/hl2ss
20. https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/?view=mrtkunity-2022-05