博士論文

Study on Application Systems using Smart Glasses Aimed at Field Service Assistance
（フィールドサービス支援を目指したスマートグラス応用システムに関する研究）

李　庭豪

令和 2 年度　博士論文

Study on Application Systems using Smart Glasses Aimed at Field Service Assistance
（フィールドサービス支援を目指したスマートグラス応用システムに関する研究）

指導教員　　鈴木　宏正　教授

東京大学大学院　工学系研究科　精密工学専攻

37-177237

LI, TING-HAO
李　庭豪

Abstract

Augmented reality (AR) has been a promising tool in engineering fields. AR systems can show digital contents to the worker’s view of a real scene and give understandable information and instructions. Thus, it is a promising way for field service applications such as assembling, maintenance, and construction and for training novice workers. Especially, smart glasses are a key device for such AR applications and usually have embedded sensors such as an RGB camera and a depth camera to collect data from environments. One of the prospects of smart glasses technology is to let the worker carry and use only smart glasses for such field service applications and training. To realize it, using AR glasses equipped with eye tracking sensors is a possible way and promising to construct a more complete system for field service assistance. With AR technology, the worker can directly receive computer-generated information during the operation. Simultaneously, worker′s attention can be recorded by eye tracking sensors for performance assessment and skill training. Although the development of such AR glasses with eye tracking is still in its infancy. we can establish related systems in advance by using existing smart glasses. In this research, we study two types of smart glasses to develop two prototype systems with the aim of field service assistance: (1) eye tracking glasses for user’s cognitive research and analyses and (2) AR glasses for assembly assistance.

For eye tracking glasses, which record the user’s gaze fixation in a scene video, most related applications are in 2D space, and there should be more applications in 3D space to give more understandable gaze analysis results. In this study, we propose a method to demonstrate user’s gaze in 3D space using a pair of eye tracking glasses. After a user performs an eye tracking recording in a certain environment, we generate a 3D mesh model of the scene from the frame images in the scene video by applying the image registration method. The intersection of a triangle in the model and user’s line of sight, which is determined by linking the camera center of the frame image and the recorded gaze point, is the target 3D gaze fixation. Moreover, based on this methodology, we propose another method to compare multiple users’ 3D gaze visualization more efficiently and effectively. Similarly, by applying the image registration method, we register the frame images of all the users' recording into the same 3D model generated from one of the users' recording, that is, reconstructing only one model of the scene to visualize all users' 3D gaze. In our experiment for three users observing the same scene, the processing time decreases by 50%. In addition, an eye tracking recording in a room-scale environment is conducted to demonstrate the advantages of 3D gaze visualization in complex and large-scale environments, which can be hardly demonstrated well by typical 2D visualization methods. With the developed methods, we can use only eye tracking glasses to generate user's 3D gaze visualization and compare users' gaze difference to train novice workers more efficiently and effectively.

For AR glasses, the development of AR technology has enhanced the experience of assembly operations by showing virtual parts of assembly at installation locations to a worker. Good assembly instructions can improve the effectiveness of assembling operations and training; meanwhile, it is important to detect whether misassembling occurs during such operations. In the study of AR glasses, we construct an AR assembly assistance system using only a head-mounted display of Microsoft HoloLens. Because the HoloLens is originally designed for room-scale applications, we use point clouds generated by a depth camera in the HoloLens and propose two methods to apply the HoloLens to desktop applications. One method is coordinate calibration to display virtual objects at installation locations by aligning reference virtual object, which is at the origin of the virtual world, to the associated reference real object's position. The other evaluates in real time whether misassembling has occurred by evaluating misalignment between real and virtual objects. For efficiency, we compare the depth images of the real and virtual objects instead of the calculation in 3D space. With the preliminary tests, the position error can be within ±1 cm and misalignment evaluation can be performed at 30 fps. Thus, with the presented methods, a standalone AR assembly assistance system can be realized to support assembling operations and training.

In summary, with the usage of existing smart glasses and developed methods, we develop two prototype systems for the assistance of field serves applications. The proposed 3D gaze visualization method and the AR assembly assistance system are possible to assist the worker who needs to move in room-scale environments. In the future, as the mature development of AR glasses with eye tracking, we can apply the proposed systems into such smart glasses and construct a standalone system. The system will be able to assist the worker in assembling operations through AR. Meanwhile, the worker's visual attention can be recorded during the operations, and then visualized in 3D space. The 3D gaze visualization is further used to assess the worker's skill and to compare between experienced and novice workers for training and analytic studies. This will construct a more complete system for field service assistance.

Outline

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Background

Augmented reality (AR) technology has been a promising way for the future of workers of various tasks. An AR system can superimpose computer-generated information onto the user's view of a real-world scene. With the development of AR technologies, AR applications have been commonly used in our life, such as entertainment, education, design, and so on. Such AR systems also play an important role in *field service applications*, such as equipment installation, maintenance, construction, assembly, etc. [1][2] because they allow the workers to access task information easily as they are in the working field. With augmented information shown to a technician through a head-mounted display (HMD) or a tablet, the technician can receive information and indication immediately when they service complex equipment [3] as shown in Fig. 1-1. This increases the effectiveness and efficiency of field service work. For manual tasks, it has been shown that the usage of AR systems improves task performance and decreases error rate [4][5].

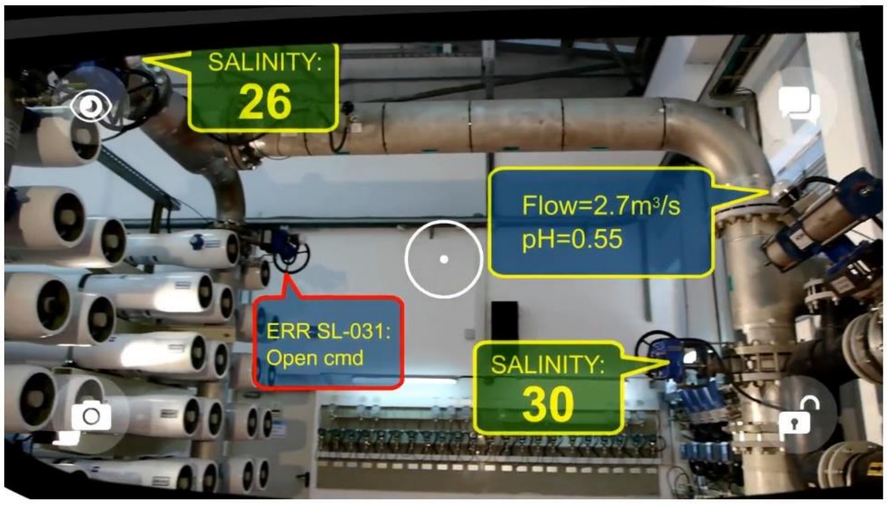


Fig. 1-1 AR-based field service knowledge software platform provided by Fieldbit [3]. Computer-generated information is fused into the real scene to show the current status of the equipment.

In addition to the assistance to expert workers, using AR technology is a possible way to train novice workers to acquire new assembly and maintenance skills [2][6][7], as shown in Fig. 1-2. Since assembly and maintenance tasks can be quite complicated, it is challenging to train novice workers efficiently and effectively. Compared with trainings such as one-on-one training with an experienced workers and self-learning from videos or text documents, using AR-based training systems may support the trainings to a large amount of novice technicians at the same time with the assistance of visualized operating indications from the system.

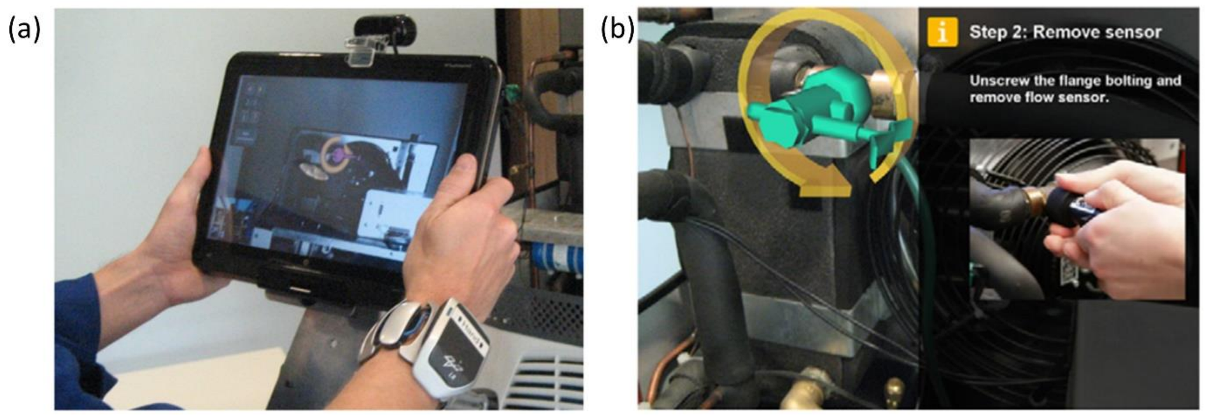


Fig. 1-2 AR training platform: (a) A tablet to inspect machinery and digital indications and (b) visual aid with virtual objects [2].

AR glasses and tablets (or smart phones) are two main types of devices for AR applications. For tablets and smart phones, which are usually equipped with an RGB scene camera, as shown in Fig. 1-2, computer-generated information is added into the recorded video of the real-world scene. Besides equipped with an RGB camera, it is a tendency to add a depth sensor together to expand augmented reality capabilities of the device. For example, Apple iPad Pro 2020 (tablet) and Samsung Galaxy Note 10+ (smart phone) have a depth camera to measure the distance of surrounding objects and to expand AR experiences, such as 3D scanning and interaction with real-world objects. However, although tablets and smartphones have been widely used in our daily life, they may not be appropriate AR devices for manual tasks because the user has to hold the devices and can hardly perform the tasks.

On the other hand, for AR glasses, it is an eyeglasses-type of AR devices, which are equipped with see-through lens to display digital information in front of user's eyes. Because AR glasses make the user's hands free, such smart glasses can be a key device for AR-based field service applications. For example, in Fig. 1-3, a company called Trimble developed an AR assembly assistance system for construction fields. Through AR glasses, workers can see virtual pipelines displayed at the installing locations. This kind of AR-based assembly instruction gives the worker a complete image about the target assembly. Moreover, advanced smart glasses are equipped with various sensors such as a depth camera and microphones to provide functionality of constructing digital data of the real world and voice commands to operate the AR system, respectively.

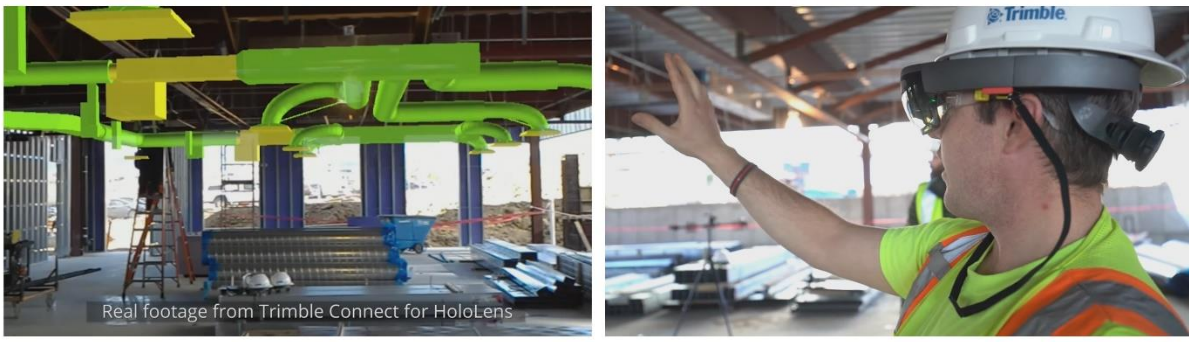


Fig. 1-3 Observe virtual assembly through AR glasses in a construction field [8]

Furthermore, besides the assistance during operations, reviewing and evaluating worker's performance in the operations is important to improve worker's ability. Hence, skill assessment and training are necessary parts to assist in field service applications. Eye tracking can be a quantitative tool to provide suitable metrics [9][10] for skill assessment and training. By using eye tracking devices, we can measure worker's visual focus of attention, which is highly correlated with where the worker is focusing attention. For example, Hasanzadeh et al. [11] used eye trackers for safety training in construction fields. Workers were required to observe images of construction fields and to find potential hazard situations. Meanwhile, the workers' visual attention on the images was recorded, analyzed, and then displayed in the form of heatmaps. Fig. 1-4(b) shows that less experienced workers are more stimulus driven and focus on imminent hazards. On the other hand, Fig. 1-4(c) shows that more experienced workers are more goal driven and have a balance in focusing and dividing their attention across the scene. In comparison with more experienced workers' results, it can improve novice workers' hazard-detection skills. Thus, eye tracking is a promising tool to train and evaluate worker's skills.

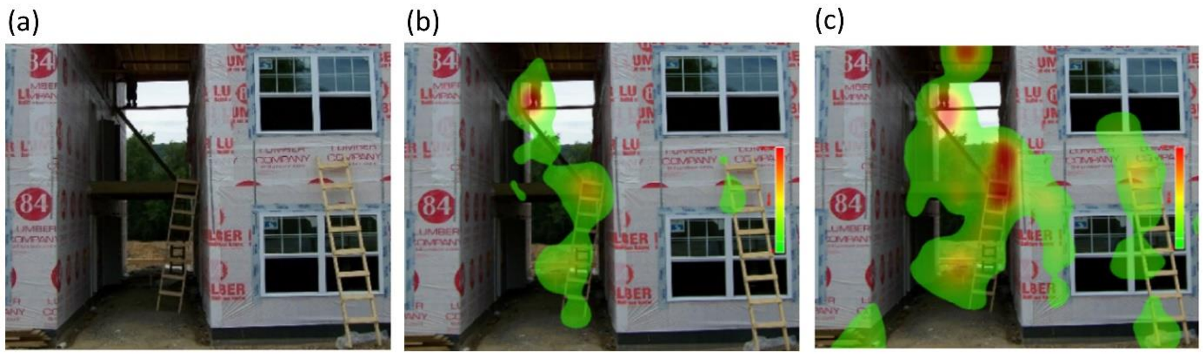


Fig. 1-4 Measuring impacts of safety knowledge on construction workers using eye tracking technology: (a) original picture, (b) attentional distribution for the group of less experienced workers (< 5 years), and (c) attentional distribution for the group of more experienced workers (> 10 years) [11].

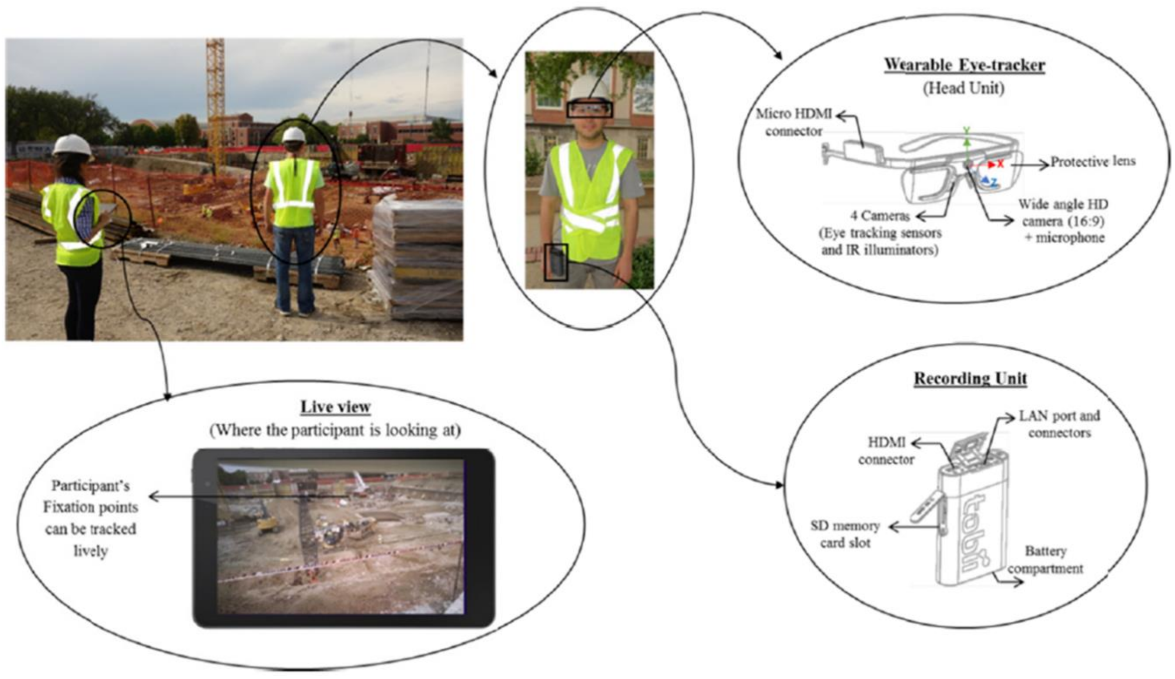

Wearable Eye-tracker
(Head Unit)
Micro HDMI connector
Protective lens
4 Cameras (Eye tracking sensors and IR illuminators)
Wide angle HD camera (16:9) + microphone
Recording Unit
LAN port and connectors
HDMI connector
SD memory card slot
Battery compartment
Live view
(Where the participant is looking at)
Participant's Fixation points can be tracked lively

Fig. 1-5 Record the worker's visual attention by eye tracking glasses when the worker is working in the field [9].

To capture workers' natural behavior when they perform the task in the field, using eye tracking glasses, which is a mobile eye tracking device, is better than using the fixed type of the eye tracking device. Because this wearable device can let the worker walk around environments, the worker who wears the eye tracking glasses can keep working as usual, and the worker's working condition can be recorded and analyzed for skill assessment and training. Fig. 1-5 shows the usage of eye tracking glasses to track a worker's attention in a construction field.

In summary, with the portability and various functions, smart glasses can be a key device for assisting the worker engaged in filed services in the future. Different technologies in smart glasses bring different aspect of assistance. The AR system can provide information to the worker during the operation and help the worker complete the task more effectively and efficiently. Moreover, eye tracking technology can record the worker's performance, and the eye tracking data can be utilized to improve worker's skills such as hazard-detection skills.

1.2 Motivation and Objective

In the working field, smart glasses can make the worker's hands free and usually have embedded sensors to collect data from the environment to provide various functionality. With those features, one of the prospects of smart glasses is to let the worker carry and use only smart glasses for such field service applications and training. Moreover, since smart glasses are wearable devices and can let the worker move around the working field, which should be a room-scale or larger environment, it will be beneficial to develop assistant systems for working in such environments. Therefore, in this work, we mainly focus on indoor field service applications and the development of AR-based field service assistance systems for indoor environments.

Fig. 1-6 shows the designed workflow of our target AR-based field service assistance system. The system is implemented in smart glasses and provides two main functions. One is to assist the worker during operations through information overlay of AR technology. Computer-generated information is displayed to the worker's view of the real scene to directly give operating information, such as instructions and status of the operation. The other is to record worker's visual focus of attention during the operations through eye tracking technology. We can then visualize and analyze the worker's gaze data for skill assessment and training.

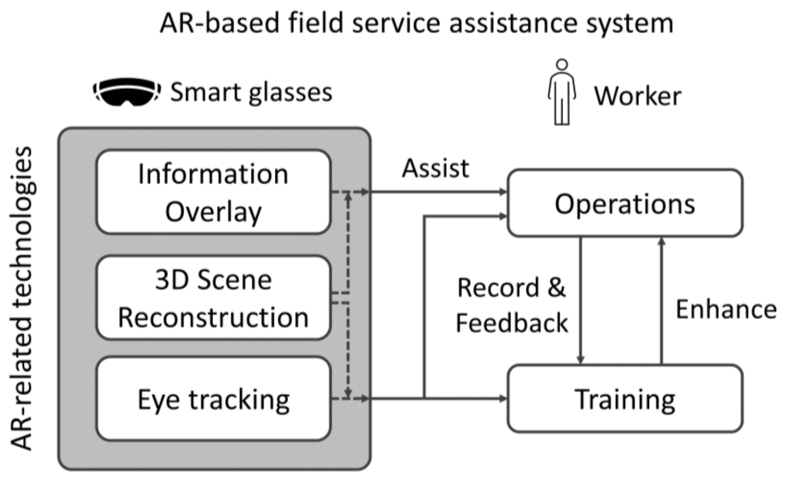


Fig. 1-6 A schematic plot of the workflow of our target AR-based field service assistance system

To realize a such system for room-scale environments, in the aspect of technology, we need three kinds of technologies at least: (1) information overlay, (2) 3D scene reconstruction, and (3) eye tracking. To assist the worker during operations, we need information overlay which is related to AR technology to provide indication, and we also need the 3D structure of the environment to detect real objects in the environment for the AR applications such the evaluation of misassembling in assembling operations. Moreover, to train the worker, we need eye tracking technology to track worker′s visual attention, and we also need a 3D model of the environment to visualize worker′s 3D gaze fixations which is beneficial to show the room geometry for indoor field service applications.

In the aspect of hardware, using AR glasses equipped with eye tracking sensors is a possible way to construct an AR-based field service assistance system. It contains AR and eye tracking technologies. With the integration of eye tracking technology, such smart glasses can not only enable better user experience in AR, such as better overlay accuracy of virtual objects and hands-free interaction with virtual objects, but also monitor worker's performance for skill assessment and training [12][13]. However, the development of such AR glasses with eye tracking sensors is still in its infancy in 2020. Nevertheless, we can use existing smart glasses to develop related systems for AR-based field service applications and training.

In this study, we use and investigate two types of smart glasses to develop two prototype systems with the aim of field service assistance: (1) AR glasses (AR HMD) and (2) eye tracking glasses as shown in Fig. 1-7. For AR glasses, we focus on AR assembly assistance during assembling operations which contains technologies of information overlay and 3D scene reconstruction. For eye tracking glasses, we mainly focus on the issue of 3D gaze visualization which contains technologies of eye tracking and 3D scene reconstruction. The two systems that cover the three technologies serve as the subsystems of the AR-based field service assistance system. Although 3D scene reconstruction is the common part of the two systems, by considering to the sensors embedded in the two pairs of smart glasses, we may have to use two kinds of 3D scene reconstruction methods for the two prototype systems.

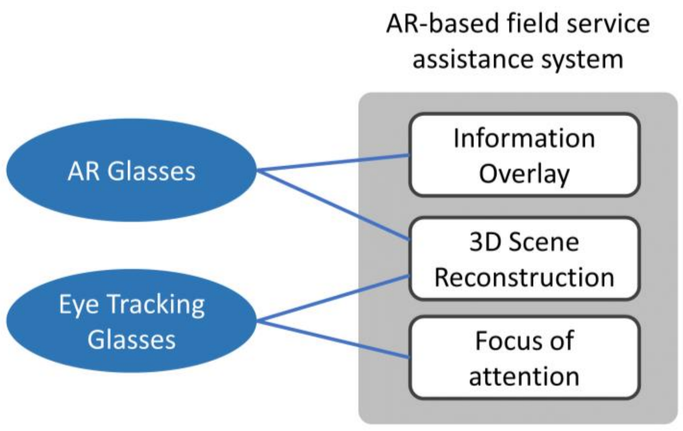


Fig. 1-7 Preliminary development of an AR-based field service assistance system using AR glasses and eye tracking glasses.

1. AR glasses (AR HMD)

AR glasses are equipped with see-through lens to display computer-generated data to the user. In the field of assembly assistance, AR glasses are potential to assist the worker in assembling operations through showing virtual parts of an assembly at installing locations to the worker's view of a real scene. Besides giving instructions, it is important to prevent the worker from making assembling errors. Therefore, such assembly assistance system needs to track the condition of the assembling operation in real time. Using AR markers and external devices such as an RGB or RGB-D camera is common to display the virtual parts at target positions and to evaluate misassembling, respectively [14]. However, it means that the effective working area is limited to the sensing zone of the external camera which is fixed to somewhere in the workplace. To overcome this limitation, we use AR glasses of Microsoft HoloLens, which has embedded RGB and depth cameras, to develop an AR assembly assistance system. Simultaneously, misassembly should be evaluated and visualized by the AR glasses in real time to avoid assembling errors. The embedded depth camera can give the information of 3D scene reconstruction for misassembly evaluation.

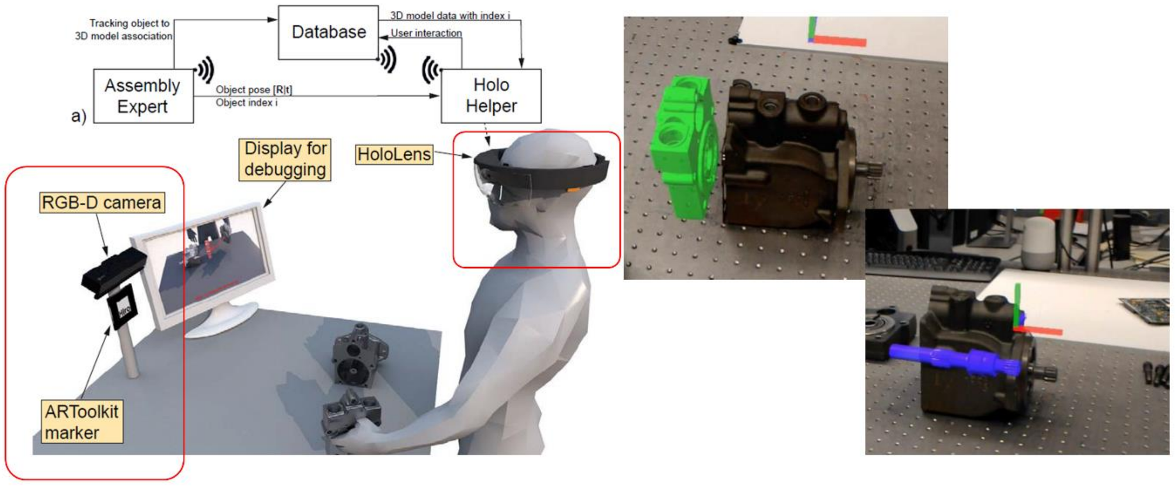


Fig. 1-8 An AR assembly assistance system using AR glasses, an RGB-D camera, and AR markers [14]

2. Eye tracking glasses

Eye tracking glasses are a mobile eye tracking device with an embedded RGB camera. When the worker who wears the eye tracking glasses works and moves around the workplace, worker's gaze fixations will be tracked and marked in a recorded scene video. To do statistical analysis, the analyzed eye tracking result such as heatmap is usually displayed in a panorama or multiple images of the scene, including multiple views of the environment. To better demonstrate the eye tracking result of the 3D environment, displaying user's gaze fixation in a 3D model of the scene is a way which can show the multiple views and the geometry of the environment in one model. From each worker's eye tracking recording, we can reconstruct the 3D model of the scene from the scene video, which records both the user's visual attention and the views of the environment, by image-based 3D reconstruction. Furthermore, to compare the eye tracking results between different workers, who perform the same work in the same environment, for the evaluation of their performances, it is appropriate to share the same 3D model of the scene to visualize their gaze fixations. This will give the same criterion for the assessment of their gaze results. In this study, we use eye tracking glasses of Tobii Pro Glasses 2.

To sum up, in this work, we focus on the issues of AR and eye tracking technologies of smart glasses for AR-based field service assistance systems in indoor environments, respectively. Because smart glasses allow the worker to move around the workplace, we need appropriate systems for such situation. By using provided functionality and sensors in the smart glasses, we aim to develop the two prototype systems through AR glasses and eye tracking glasses:

AR glasses (Chapter 2)

- Develop an AR assembly assistance system with the evaluation and visualization of misassembly in real time.

Eye tracking glasses (Chapter 3)

- Visualize and compare multiple users' 3D gaze visualization more efficiently and effectively by sharing the same 3D model of the scene.

1.3 Thesis Structure

Fig. 1-9 shows the thesis structure, which contains four chapters. Chapter 1 introduces the background, the objectives, and the structure of the work. With the aim of AR-based field service assistance by using smart glasses, we divide the research work into two parts as introduced in Sec. 1.2: (1) the AR assembly assistance system and investigate their own issues and (2) the 3D visualization of eye tracking results in Chapters 2 and 3, respectively.

Chapter 2 concentrates on the AR technology, and we use AR glasses as the target smart glasses. With the embedded sensors and functionality provided by the AR glasses, we develop an AR assembly assistance system and focus on the issues of coordinate calibration and misassembly detection in real time, which are import for the correct displayed location of virtual objects and the avoidance of assembling errors, respectively. We then design and conduct experiments to verify the system performance.

In Chapter 3, we focus on the eye tracking technology and use eye tracking glasses as our test device. We first develop a method of constructing single user's 3D gaze visualization from an eye tracking recording. To further compare with other users' 3D gaze results, we propose a more effective and efficient way to reduce the generation time of multiple users' 3D gaze visualization. Eventually, we verify the proposed 3D gaze visualization system and conduct a room-scale experiment to demonstrate the advantages of 3D gaze visualization.

Eventually, Chapter 4 is the conclusion of the research work. We sum up the contribution of our research work, discuss the potential problems in each topic, and describe the possible ways of improving the systems and the future work.

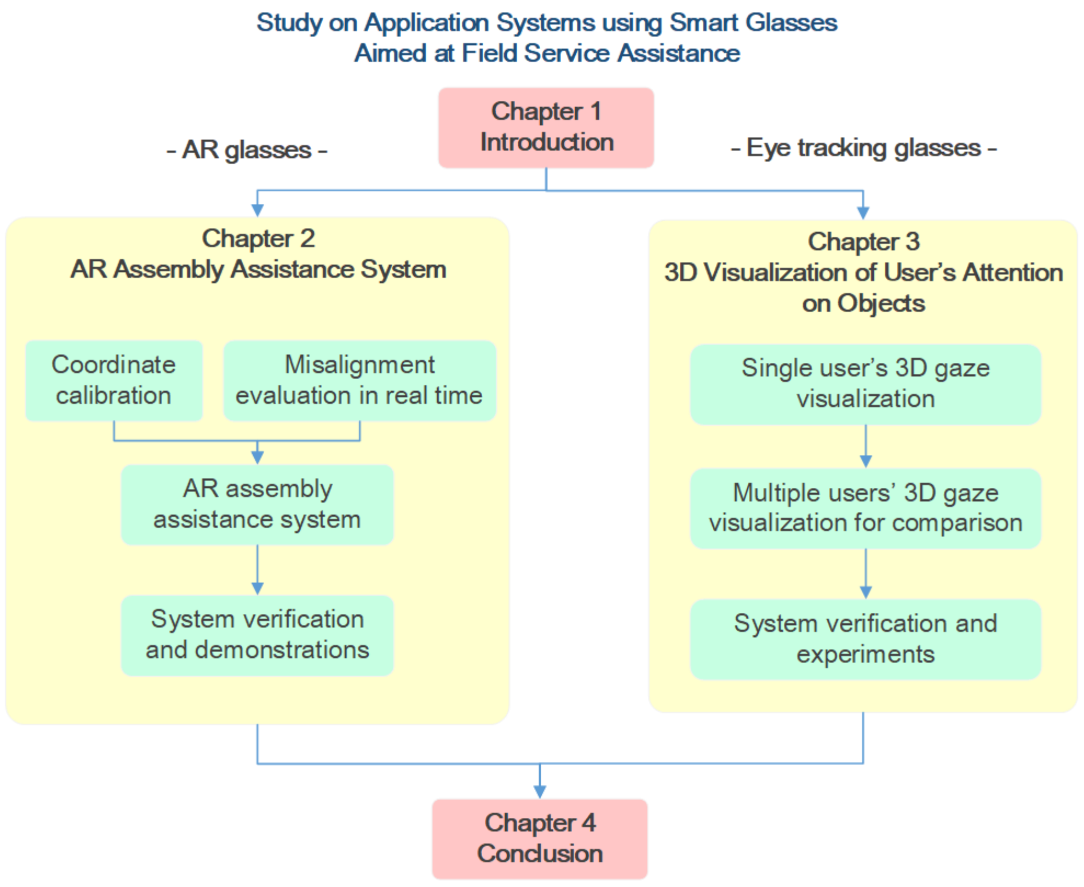

Study on Application Systems using Smart Glasses
Aimed at Field Service Assistance
Chapter 1
Introduction
- AR glasses -
- Eye tracking glasses -
Chapter 2
AR Assembly Assistance System
Coordinate calibration
Misalignment evaluation in real time
AR assembly assistance system
System verification and demonstrations
Chapter 3
3D Visualization of User's Attention on Objects
Single user's 3D gaze visualization
Multiple users' 3D gaze visualization for comparison
System verification and experiments
Chapter 4
Conclusion

Fig. 1-9 Structure of the research work

Chapter 2
HMD-Based AR Assembly Assistance System with Efficient Evaluation of Misalignment between Real and Virtual Objects

2.1 Introduction

In this chapter, we first study the main assistance part of the AR-based field service assistance system during operations, and we focus on AR assembly assistance to assist the worker in assembling operations using AR glasses.

Augmented reality (AR) technology [15][16] has broad application in the world. It has been used in various fields, such as gaming [17] and medical training [18], and education [19]. The core concept of AR is to put an overlay of 3D virtual objects into real scenes as the virtual objects really exist in the physical world. In comparison with virtual reality (VR) technology, which constructs a complete virtual world by a computer, AR applications put great emphasis on the connection between the real and virtual worlds. Therefore, it needs the understanding of real environments to some extent.

In engineering fields, assembly instruction is a potential application of AR [20]. Through a head-mounted display (HMD) or a tablet, an AR system can give visualized instructions, in the form of texts, symbols, and assembly animations, in front of an operator to make the operator easily understand and follow the instructions. For instance, the AR system can display a computer-aided design (CAD) model of a component from an assembly at its installation location to lead the operator to complete an assembly operation step by step. For complex assemblies, it is beneficial to provide the operator concise and effective instructions for efficient assembly operation and training instead of relying on that operator to read a manual during the assembly operation. In addition, using an HMD is more convenient than using a tablet or a smartphone because the HMD lets the user's hands free.

An AR-based assembly assistance system must be able to show CAD models (virtual objects) at their installation positions in the real world for reliably giving instructions. That is, the system should determine precise transformation relationships between coordinate systems, such as the coordinate systems of the real world, of the virtual world in which the CAD models are defined, and of the user's position. This is commonly realized using AR markers [21] to evaluate transformation matrices between various coordinate systems. In these arrangements, in addition to the HMD, several external devices, such as AR markers and an RGB-D camera, are typically prepared and set up before the operation. This limits the working environment and increases the time taken to set up the instruments. In addition to assembly instruction, it is important to evaluate in real time whether there is an occurrence of misassembly during the operation [22]. Prevention of assembly errors can avoid unexpected increases in assembly time or serious damage to the assembly product. Thus, it is critical to evaluate whether the real object is placed in the target position to confirm a reliable transition to the next assembly stage.

In this study, we aim to use an AR HMD, Microsoft HoloLens, as the main device establishing a basic AR-based assembly assistance system that can display CAD models to the system's user for assembly instruction and can simultaneously evaluate any possible occurrence of misassembly. Microsoft HoloLens contains eight environment sensing cameras and a depth camera to position itself in the real world and to construct meshes of the physical environment, respectively. With these features, we propose two methods to provide functions for the system: coordinate calibration and efficient evaluation of misalignment between the real and virtual objects.

First, coordinate calibration is to determine a transformation relationship between the real and virtual worlds to make the virtual objects displayed in the desired working area. We realize coordinate calibration by rough alignment and precise alignment to transform a reference virtual object, which is at the origin of the virtual world, to the position of the corresponding reference real object, which defines the working area. These two alignments are realized by user's hand manipulation and the point-to-plane iterative closest point (ICP) method to gradually determine the transformation. To apply the point-to-plane ICP, instead of using the HoloLens-generated meshes of the real object, we use the original point cloud data to represent the digital content of the real object. Because the resolution of HoloLens-generated triangles varies from a few centimeters to dozens of centimeters, which is so large to give a great influence on the result of precise alignment particularly in the case of desktop applications. We first aim at such desktop applications, while the standard applications of the HoloLens are of room scale for objects such as furniture larger than a cube with an edge of 30 cm. One of the contributions of the research is to adapt the HoloLens to the desktop applications by using the point cloud to achieve precise alignment in a desktop area.

Second, to evaluate misassembling by misalignment of parts in 3D efficiently, we compare depth maps of the real and virtual object to achieve misalignment evaluation between the real and virtual object in real time that is computed by the poor computation resource of the HoloLens. We also study different methods to determine an effective way for misalignment evaluation. With preliminary experiments, we use the centered cosine similarity method as an indicator to evaluate whether there is any occurrence of misalignment between the real and virtual object with the accuracy of approximately ±1 cm.

In summary, our research objectives in this study contain

- Use only HoloLens to develop an AR assembly assistance system
- The system can perform desktop applications (overcome target applications of the HoloLens for room-scale environments)
- Develop methods to evaluate misalignment between real and virtual objects
    - at real time rate on the HoloLens
    - within the accuracy of ±1 cm in the case of desktop applications.

We aim at developing an AR assembly assistance system with these features.

With the two proposed methods, preliminary demonstrations show that we can use only an AR HMD, Microsoft HoloLens, to construct an essential AR-based assembly assistance system with evaluation of misalignment between real and virtual objects in real time. Moreover, the system can be applied on a desktop area and overcomes the limit of the HoloLens, which is originally used for room-scale environments.

In this chapter, in Section 2.2, we first review the related work on AR-based assembly assistance systems and applications of Microsoft HoloLens. Section 2.3 introduces the design of our system and detailed information of HoloLens. Methods of coordinate calibration and misalignment evaluation are described in Sections 2.4 and 2.5. Implementation and demonstrations of the proposed system are showed in Section 2.6, and Section 2.7 is the summary of this study.

2.2 Literature Review

Several studies have focused on AR-based systems by using various methods to assist assembly operations. To show CAD models of parts at correct locations in a real scene to the operator, first of all, it is necessary to confirm that the system can determine an accurate transformation matrix between the real and virtual worlds. Using AR markers is a common way to find the transformation relationship. In Fig. 2-1, Sääski et al. [23] set AR markers around the assembly to track the pose of the HMD relative to the markers. Once the relative pose of the HMD is known, the CAD model can be displayed on or near to the markers.

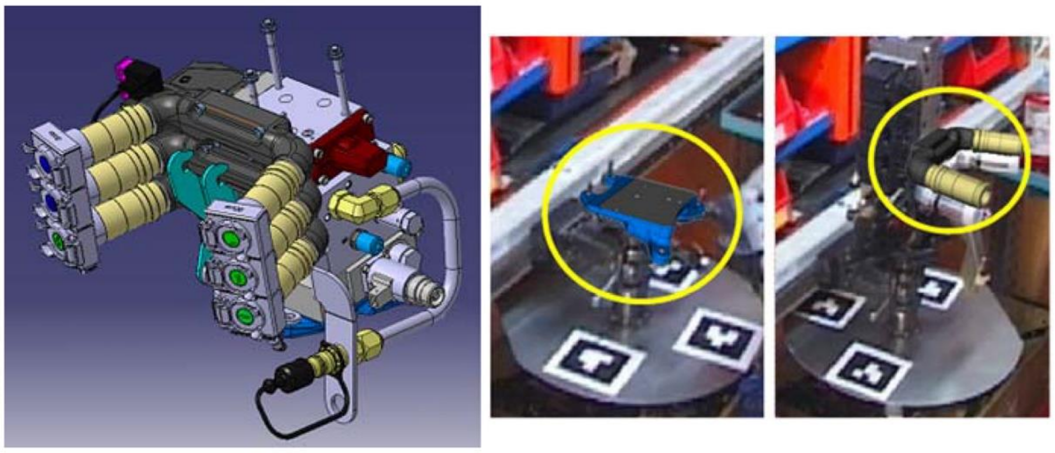


Fig. 2-1 3D CAD model of finished assembly (left) and display of the next component as a virtual part to the user (center and right) [23]

Moreover, in addition to essential functions for assembly instruction, it will be helpful if the system can monitor assembling operation. Adding external devices such as RGB or RGB-D cameras into the system is common to monitor assembly operation and to evaluate whether there is misassembly. In Fig. 2-2, Mura et al. [21] used an RGB camera to track positions of real objects and to evaluate whether there is an occurrence of the assembly error. This verification provided better training to the operators and offered the potential to prevent serious damage to products. On the other hand, Alves et al. [22] used an RGB-D camera to detect the assembly errors. Radkowski et al. [14] used an RGB-D camera to track a real object inside the sensing zone of the camera and made a CAD model that could follow the real object. However, using multiple devices may limit the portability and working area of the system. Real parts of an assembly can only be tracked when they are inside the sensing zone of the camera, which is fixed somewhere in the working area.

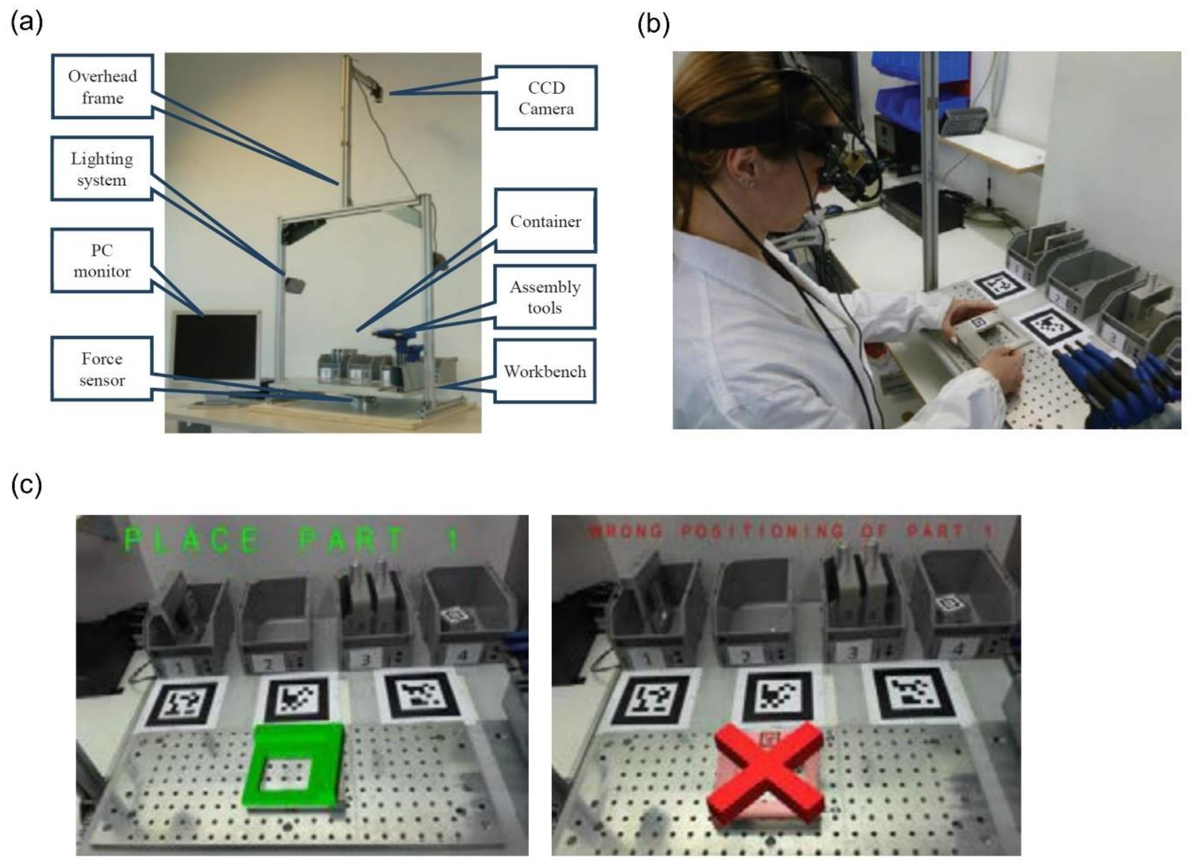


Fig. 2-2 AR assembly assistance system of Mura's study [21]: (a) assembly workstation, (b) observe virtual parts through an HMD, and (c) visual messages for detecting assembly errors.

Adding sensors, such as RGB or RGB-D cameras, onto the HMD is a possible way to make the system portable and overcome the limited range of the working area. Microsoft HoloLens is one of commonly used commercial HMDs, which has an embedded depth camera to scan the real world. Evans et al. [24] used the HoloLens to construct a system that performed assembly operations in a room-scale environment. However, although HoloLens can scan the real world by using the embedded depth camera and generate meshes of a scene, the meshes are not detailed enough to track most physical parts smaller than furniture and support an assembly application. Instead, AR markers were placed in the environment and used to precisely define the position of a real object.

Because the original usage of HoloLens is for applications in room-scale environments, using HoloLens-generated rough meshes is beneficial to be used to detect large obstacles and planes, such as floors and walls, in the environment. However, with the open of research mode, a system configuration to access raw data of embedded cameras in HoloLens, we are able to obtain raw depth images from the HoloLens [25], and further turn them into 3D point clouds to detect real objects.

In summary, although AR technology can make assembly operations more effective and efficient, using external devices limit the working area's scalability and the user's movement in the environment, and it may take time to perform initial device setup. These issues motivated us to develop a system that uses only an HMD, Microsoft HoloLens, to achieve the functions of assembly instruction and verification. This single device can make the system more flexible to various scales of environments and products.

2.3 Design of HMD-Based AR Assembly Assistance System

2.3.1 System Structure

Our target system allows a user to assemble parts in their installation locations in an assembly. In Fig. 2-3, the installation locations are specified in a CAD model that is supplied for the assembly and defined in the coordinate system of the virtual world. After transforming the CAD model of the assembly to an appropriate position in a working area of the real world by a transformation $\mathbf{T}_{RW \leftarrow VW}$, the user can place the real parts to the positions of the matched CAD models to complete the assembly as indicated. In addition, during the assembling operation, the system tells the user how closely a real part is aligned to its corresponding CAD model while the user was trying to place the real part in the assembly.

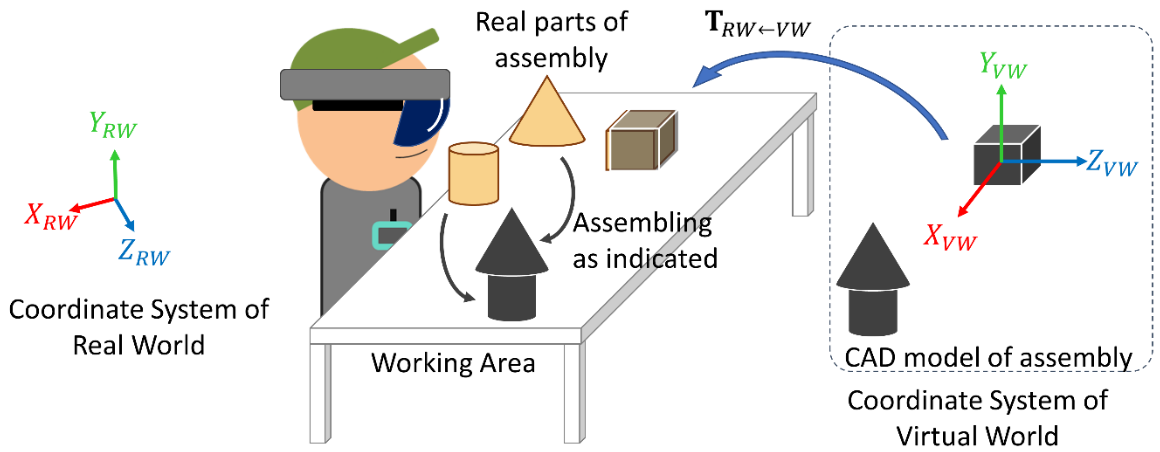


Fig. 2-3 A schematic plot of the relationship between physical parts and CAD models of an assembly

Fig. 2-4 shows the proposed system's process. First, the user needs to perform coordinate calibration and determine the transformation relationship $\mathbf{T}_{RW \leftarrow VW}$ between the virtual and the real worlds to display the CAD model in the working area. Next, the system starts to show the CAD model at its installation location to let the user understand which real object should be chosen and where it should be placed at each assembly stage. During the assembly operation, the system evaluates whether misalignment between the real object and the CAD model has occurred and visualizes the evaluation results to the user. The process continues until the assembly operation ends. Coordinate calibration and evaluation of misalignment are the cores of the system and are described in Sections 3.4 and 3.5, respectively.

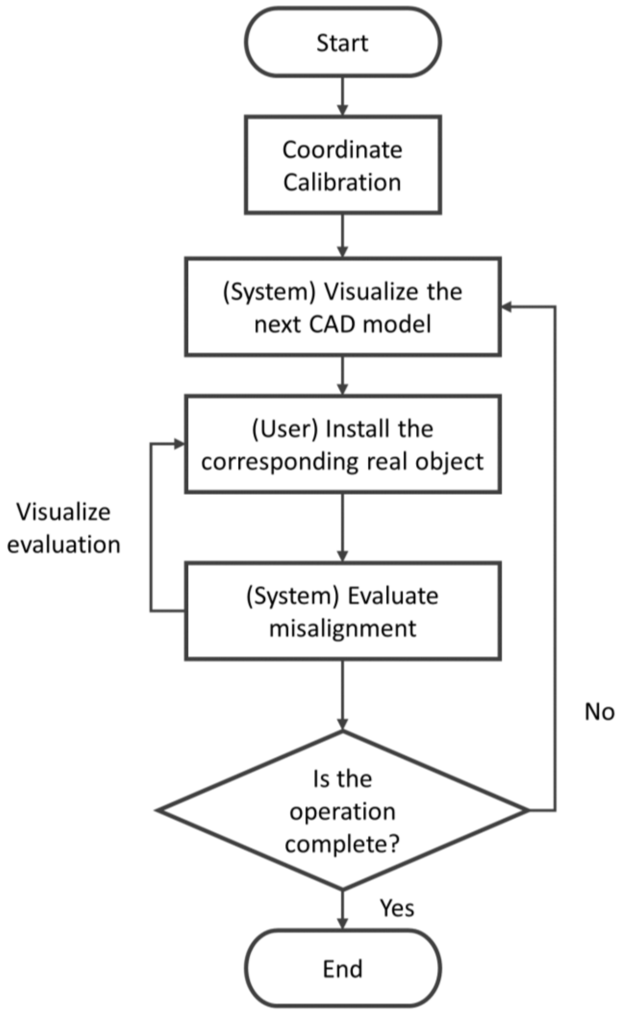

Start
Coordinate Calibration
(System) Visualize the next CAD model
(User) Install the corresponding real object
Visualize evaluation
(System) Evaluate misalignment
No
Is the operation complete?
Yes
End

Fig. 2-4 Process of the AR-based assembly assistance system

2.3.2 Mixed Reality Head-Mounted Display – Microsoft HoloLens

In this study, we use Microsoft HoloLens, which is a pair of mixed reality glasses manufactured by Microsoft. In this study, HoloLens serves as an HMD and is used for the entire computation. Moreover, we use the first generation of HoloLens in the whole study, which is referred to as HoloLens (1st gen) in general. Fig. 2-5 shows the appearance and embedded sensors of HoloLens, and more detailed device specifications [26] related to this study are shown in Table 2-1.

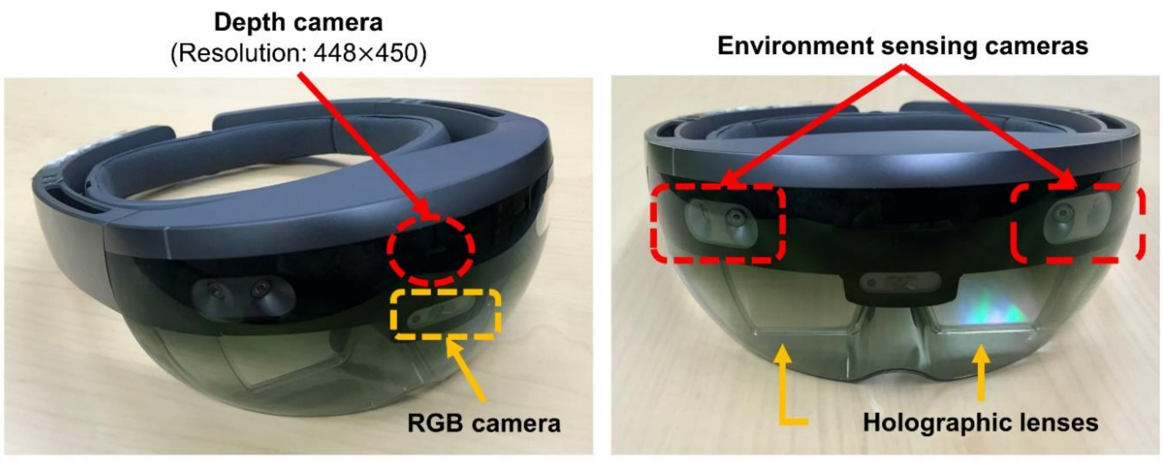


Fig. 2-5 Microsoft HoloLens (1st gen)

Table 2-1 HoloLens specifications

Weight	579g
Display	2.3-megapixel see-through holographic lenses Automatic pupillary distance calibration
Sensors	1 IMU (Accelerometer, gyroscope, and magnetometer) 4 grayscale environment sensing cameras 1 depth camera with a 120°×120° angle of view 1 2.4-megapixel photographic video camera
CPU	Intel 32-bit architecture (1GHz)
GPU	Microsoft Holographic Processing Unit (HPU 1.0)
Memory	2GB RAM
Storage	64GB
OS	Windows 10
Development tools (used in this study)	Unity & Visual Studio 2017 & C# programming language

The sensors in HoloLens collect the information of the physical world, and the HoloLens system analyzes and applies the information for various applications. For example, the system uses the four environment sensing cameras to determine where a user is in the real world, the RGB camera to record the user's view, and the depth camera using a time-of-flight technique to scan surfaces of real objects. The depth camera uses active infrared (IR) illumination to determine depth trough time-of-flight technique [27]. Thus, for the system of AR glasses, we use the embedded depth camera as the 3D scene reconstruction method. With the captured information of the real world, the user can see virtual objects through the holographic lenses as if they exist in the real world. In addition, with the Windows 10 OS and the computation-related hardware, HoloLens can act as a standalone computer and handle multiple tasks concurrently. This feature lets the user develop various applications with complicated mathematical computations.

When HoloLens starts executing a developed application, it defines a real-world coordinate system for the application, and the origin of the coordinate system in the real world is at the initial location of the HoloLens where the application starts up. This real-world coordinate system defines the user's position and virtual object positions in the real world to demonstrate the correct placement of the virtual objects to the user. More details are described in Sec. 2.6.1. Fig. 2-6 shows photos of mixed reality experience. The system shows the CAD model of a bear statue somewhere in the real world according to the real-world coordinate system. The system can display correct views of the virtual bear according to the user's location. Thus, for the user, the virtual object looks fixed in the real world.

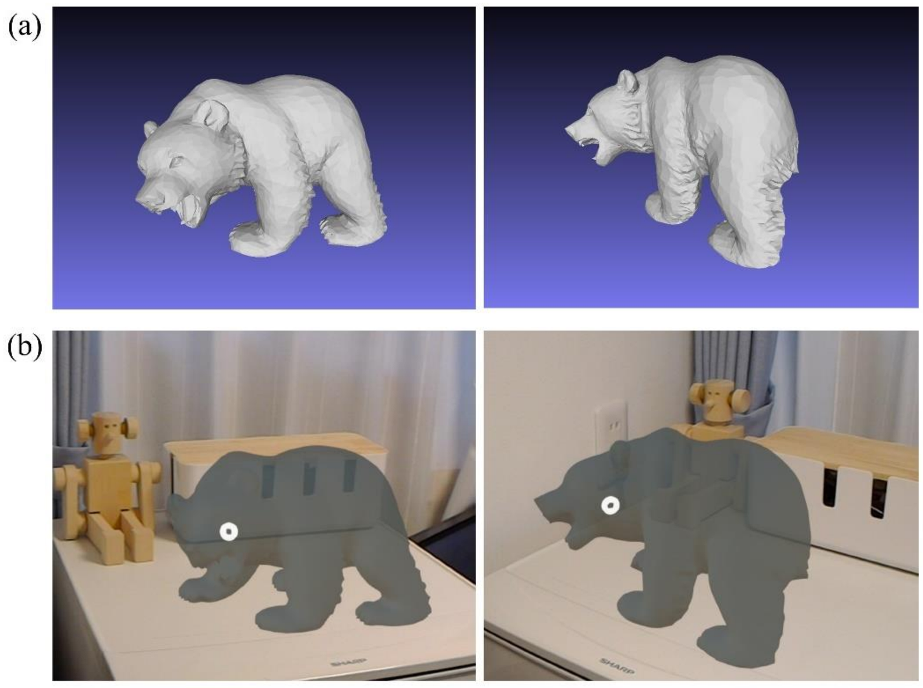
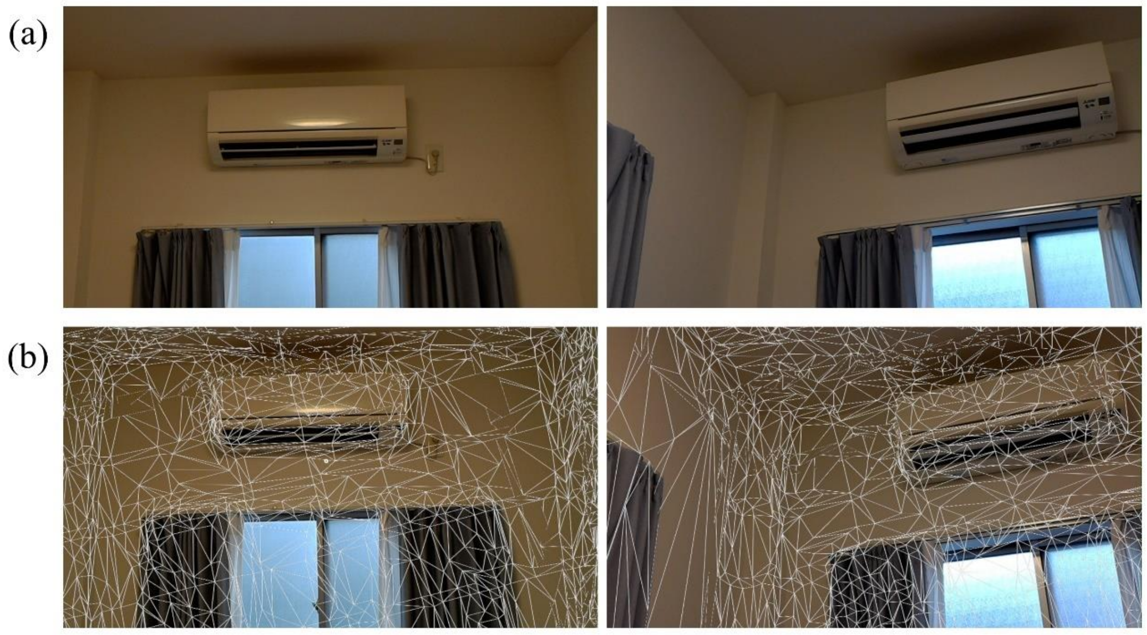


Fig. 2-6 Observe a virtual object through HoloLens: (a) the CAD model of a bear statue and (b) mixed reality photos.

There are some built-in functions in HoloLens, such as gesture input and mixed reality capture. Mixed reality capture lets users capture the mixed reality experience as a photo or video. It combines the output of the right eye′s holographic lenses with the RGB camera. We also use this function to capture user′s AR experience.

In addition, Microsoft develops and provides the Mixed Reality Toolkit (MRTK) [28] to help HoloLens developers develop various applications fast. For example, the "Two Hand Manipulatable" function allows the user to control the position and orientation of a virtual object by user′s hands. The "Spatial Mapping" function in MRTK allows the developer to obtain meshes of real-world surfaces in the environment around the HoloLens, which are automatically generated by HoloLens. As shown in Fig. 2-7, the HoloLens scans a room including an air conditioner and windows with curtains and generates the mesh of the scene, a set of triangles. The HoloLens-generated meshes are useful to describe planes, such as walls and floors, and large-scale objects or environments larger than a cube with an edge of 30 cm.



(c)

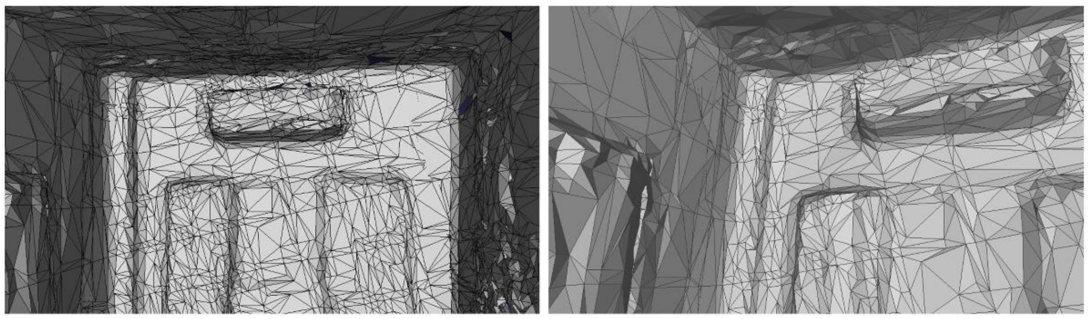


Fig. 2-7 Meshes of real-world surfaces generated by the HoloLens: (a) real scenes, (b) display of the scanned meshes in the real scene, and (c) extracted real-world meshes

The HoloLens generates real-world meshes from depth images given by the depth camera. The highest resolution of the generated mesh that the HoloLens can process in real time is approximately 1000 triangles per cubic meter [29]. The edge length of the generated triangle can vary from approximately 3 cm to dozens of centimeters. To obtain better digital information of objects, it is a possible way to obtain raw data of the depth camera, that is, the depth images. By enabling "Research Mode" in the HoloLens, a software setting for system configuration, we are able to access the low-level sensor stream data from the sensors in the HoloLens. In Fig. 2-8, eight sensor streams are available for users:

- **Environment sensing cameras** [from Fig. 2-8(e) to Fig. 2-8(h)] – grayscale images used for head tracking.

- **Depth image streams** – operate in two depth ranges, near and far:
  - ✓ Near-depth sensing is used for hand tracking. The image frame rate is 15 fps, and the effective depth sensing range is approximately from 0.15 m to 0.95 m. [Fig. 2-8(a)]
  - ✓ Far-depth sensing is used for Spatial Mapping to generate real-world meshes. The image frame rate is 1 fps, and the effective depth sensing range is approximately from 0.8 m to 3 m. [Fig. 2-8(c)]

- **IR-reflectivity streams** – show the IR reflectivity of real objects and are used to compute depth. Fig. 2-8(b) and Fig. 2-8(d) show the IR-reflectivity streams in the two depth ranges corresponding to the near-depth and far-depth sensing.

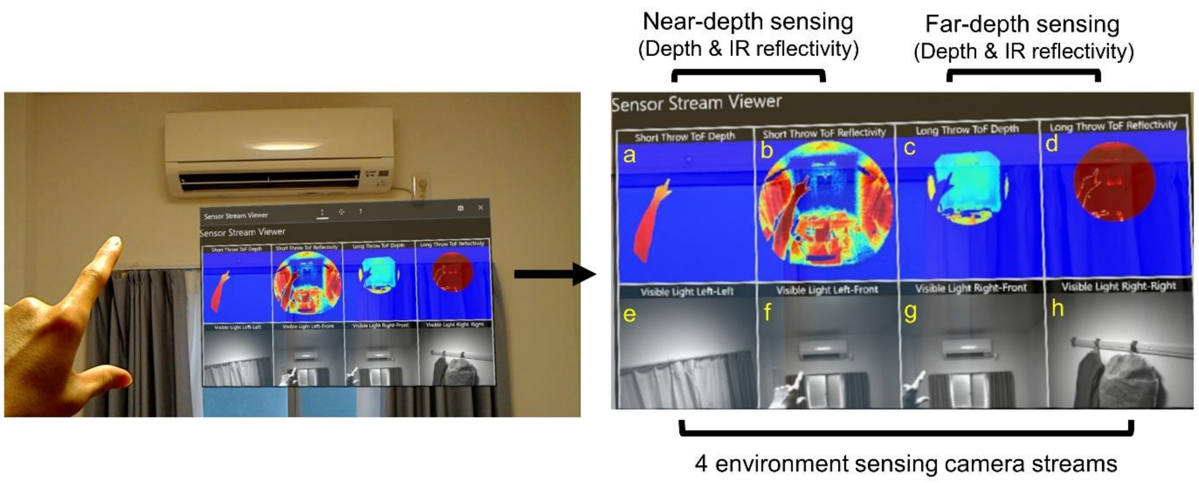

Near-depth sensing
(Depth & IR reflectivity)
Far-depth sensing
(Depth & IR reflectivity)
Sensor Stream Viewer
Short Throw ToF Depth
Short Throw ToF Reflectivity
Long Throw ToF Depth
Long Throw ToF Reflectivity
a
b
c
d
Visible Light Left-Left
Visible Light Left-Front
Visible Light Right-Front
Visible Light Right-Right
e
f
g
h
4 environment sensing camera streams

Fig. 2-8 Sensor streams available in Research Mode: (a) and (b) are the depth image and the IR-reflectivity image from near-depth sensing. (c) and (d) are the depth image and the IR-reflectivity image from far-depth sensing. (e), (f), (g), and (h) are the images of four environment sensing cameras.

Two IR illuminators in the HoloLens, operating in different frequencies, result in the two streams of the depth images at the near and far distances, as shown in Fig. 2-8(a) and Fig. 2-8(c), respectively. It is obvious that the user's hand in Fig. 2-8(a), a real object close to the user, is captured in the near-depth sensing stream, while the far-depth sensing stream can not capture it. Conversely, the near-depth sensing stream can not capture the room scene in the far distance, while the far-depth sensing can. This further shows how the HoloLens works, and allows developers to select appropriate sensor streams according to their needs.

2.3.3 Adaption of Microsoft HoloLens to the target application

The standard use of HoloLens is for room-scale environments because it is developed for mixing the real and virtual scenes, not objects. Moreover, HoloLens-generated meshes of the environment are sufficient for room-scale applications. However, objects smaller than a cube with an edge of 30 cm can not be presented well in the HoloLens-generated meshes. The error of the object's edge length is dozens of millimeters, which is too large to help us perform coordinate calibration and evaluate misalignment with the accuracy within ±1 cm. Hence, it is necessary to find other ways to acquire more precise digital information of real objects.

A. Point cloud of real objects from HoloLens

With the preliminary investigations on the HoloLens, the mesh automatically generated by the HoloLens is not a proper choice for our assembling application. Thus, we turn to acquire original point cloud data from the depth camera, which is more precise. The approach is to convert the depth image from the depth camera into point cloud in the coordinate system of the real world. Fig. 2-9 shows three important coordinate systems that the system defines and uses: (1) the coordinate system of the depth camera Σ_C, (2) the coordinate system of the user Σ_U, and (3) the coordinate system of the real world Σ_{RW}.

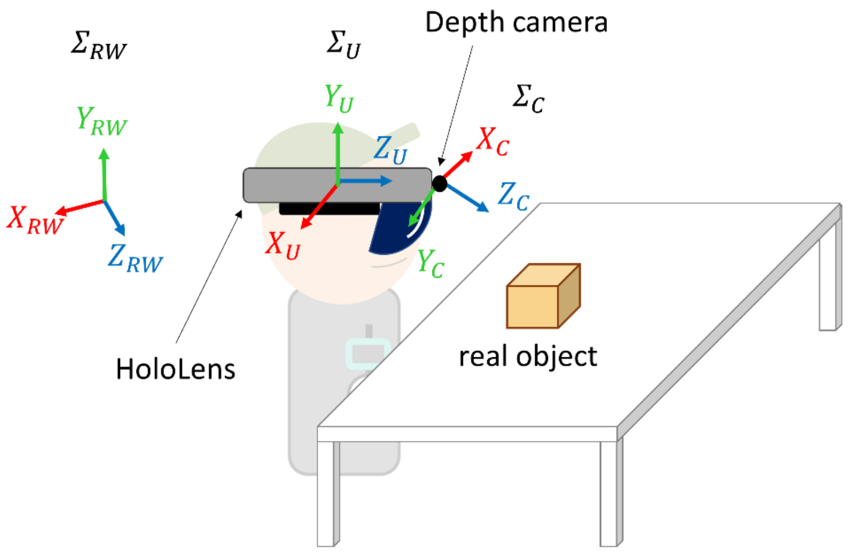


Fig. 2-9 Part of coordinate systems that the HoloLens system uses

Σ_C and Σ_U are frames of reference attached to the HoloLens and move with the HoloLens. The origins of Σ_C and Σ_U in Σ_{RW} describe the positions of the depth camera and the user in the real world, respectively. Moreover, Σ_{RW} is a stationary frame of reference with respect to the real world, and its position and orientation are the initial position and orientation of Σ_U when the application starts up. The HoloLens defines these frames of reference, and we can access the transformation matrices between them from the HoloLens to obtain the point cloud in Σ_{RW}. In Fig. 2-10, we use a cube with an edge of 10 cm to illustrate point cloud generation.

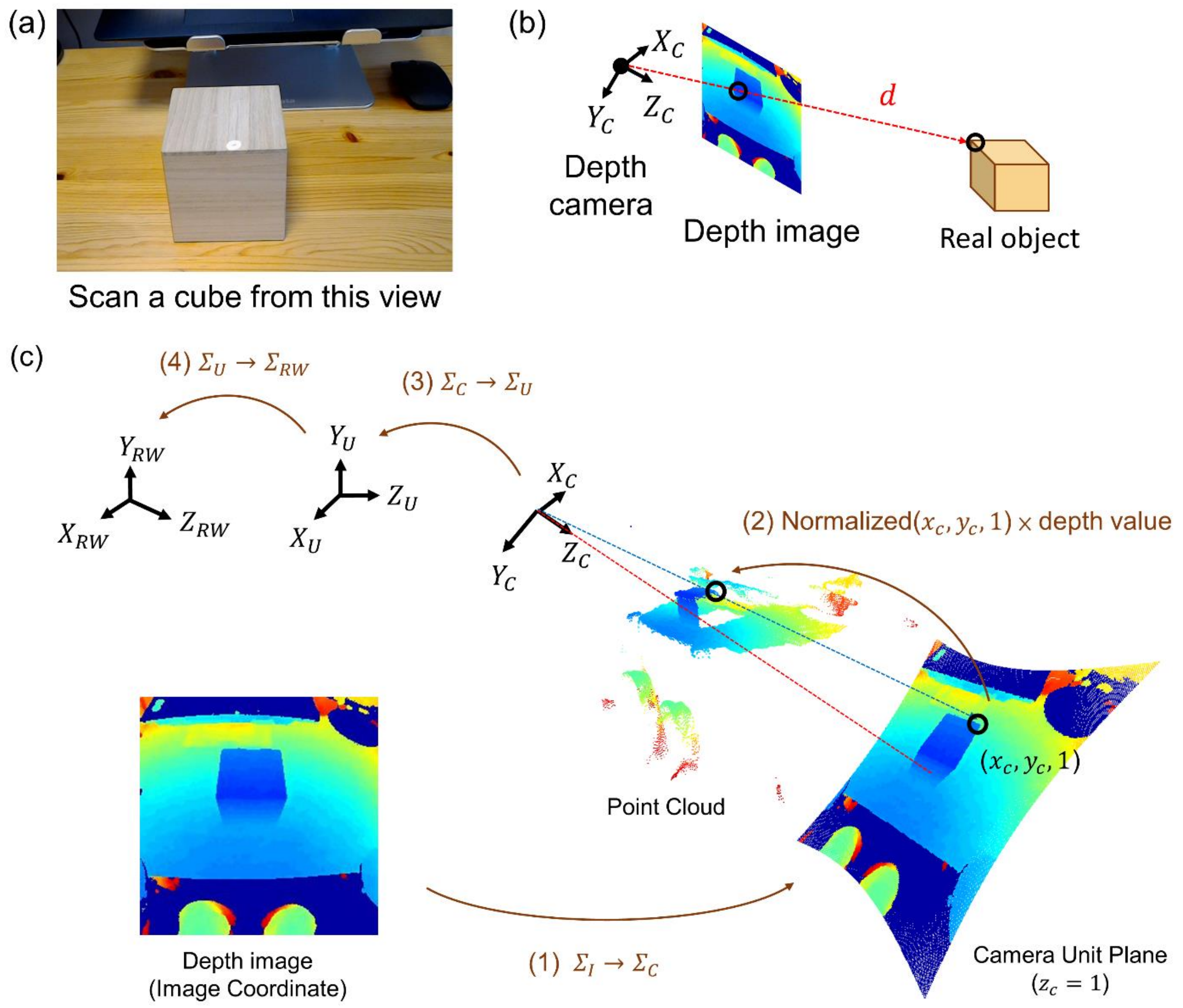


Fig. 2-10 Process of point cloud generation: (a) the view of scanning, (b) local point cloud from the depth image, and (c) real-world point cloud generation.

Notice that the depth value *d* stored in the pixel of the depth image is the distance between the depth camera center and the position of a real-scene point along the line of perspective projection; that is

$$d = \sqrt{x_c^2 + y_c^2 + z_c^2},$$

where (x_c, y_c, z_c) is the coordinate of the point in Σ_C. Fig. 2-10(c) shows the process of point cloud generation, including four steps described below:

1. Transform points from Σ_I to Σ_C: First, after obtaining the depth image from the depth camera through Research Mode, we transform the image pixels from the image coordinate system Σ_I to coordinate system Σ_C. We define

$\mathbf{M}_{C\leftarrow I}$ as the transformation that converts the representation of a point in coordinate system Σ_I into its representation in coordinate system Σ_C. Moreover, $\mathbf{M}_{C\leftarrow I}$ is a 2D-to-3D transformation with correction of lens distortion. The result after the transformation is a 3D plane, called the camera unit plane, in Σ_C, which is parallel to the $X_C Y_C$ plane and at $z_c = 1$. This transformation can be written as

$$\mathbf{p}_C = \mathbf{M}_{C\leftarrow I}\mathbf{p}_I, \tag{1}$$

where $\mathbf{p}_I$ and $\mathbf{p}_C$ are a 2D pixel point in Σ_I and a 3D point in Σ_C, respectively.

2. Obtain local point cloud in Σ_C: Next, we use the depth value information to generate local point cloud. In Eq. (1), each pixel point of the depth image is transformed to 3D point $\mathbf{p}_C$, which also describes the direction of perspective projection, and the depth value of each pixel means the magnitude of the projection. Thus, we can transform point $\mathbf{p}_C$ on the camera unit plane to position $\mathbf{p}'_C$ using the corresponding depth value *d*:

$$\mathbf{p}'_C = \frac{d\mathbf{p}_C}{\|\mathbf{p}_C\|} \tag{2}$$

where $\|\mathbf{p}_C\|$ is the Euclidean norm of $\mathbf{p}_C$.

3. Transform the local point cloud from Σ_C to Σ_U: We then convert the representation of point $\mathbf{p}'_C$ in Σ_C into its representation in user's coordinate system Σ_U using transformation $\mathbf{M}_{U\leftarrow C}$:

$$\mathbf{p}_U = \mathbf{M}_{U\leftarrow C}\mathbf{p}'_C \tag{3}$$

where $\mathbf{p}_U$ is the representation of the point in Σ_U.

4. Transform the point cloud from Σ_U to Σ_{RW}: Finally, we convert the representation of point $\mathbf{p}_U$ in Σ_U into its representation in the coordinate

system of the real world, Σ_{RW} using transformation $\mathbf{M}_{RW\leftarrow U}$:

$$\mathbf{p}_{RW} = \mathbf{M}_{RW\leftarrow U}\mathbf{p}_{U} \tag{4}$$

where $\mathbf{p}_{RW}$ is the representation of the point in Σ_{RW}, and keeps stationary in the real world.

With a series of transformation, from Eq. (1) to Eq. (4), we are able to transform the pixel point $\mathbf{p}_{I}$ in the depth image into the 3D point $\mathbf{p}_{RW}$, which is stationary in the real world, and these equations can be combined into an equation:

$$\mathbf{p}_{RW} = \mathbf{M}_{RW\leftarrow U} \cdot \mathbf{M}_{U\leftarrow C} \cdot \frac{d(\mathbf{M}_{C\leftarrow I}\mathbf{p}_{I})}{\|\mathbf{M}_{C\leftarrow I}\mathbf{p}_{I}\|} \tag{5}$$

$\mathbf{M}_{U\leftarrow C}$ and $\mathbf{M}_{RW\leftarrow U}$ are homogeneous transformations that describe both rotation and translation in 3D Euclidean space and can be accessed from the HoloLens because the HoloLens system keeps tracking transformation relationship between different coordinate systems at run time. On the other hand, $\mathbf{M}_{C\leftarrow I}$ including the information of camera intrinsic parameters and lens distortion is encapsulated as a method in the HoloLens system instead of a matrix form. To obtain the transformation result, we can invoke a low-level API called "MapImagePointToCameraUnitPlane()" [30]. The API call takes the image pixel coordinate $\mathbf{p}_{I}$ as arguments and then returns the (x_c, y_c) components of $\mathbf{p}_{C}$. More related APIs and codes for accessing point cloud can be found in *HoloLensForCV* [31], which is provided by Microsoft and contains some samples to develop applications for computer vision and robotics using the HoloLens.

B. Performance Test of the point cloud scanned by the HoloLens

After obtaining the point cloud, we conducted some scanning tests to measure the performance and the limitation of the generated point cloud and to determine appropriate objects that can be detected well during the assembling operation. First, we compare the HoloLens-generated mesh models and the point louds of objects, as shown in Fig. 2-11.

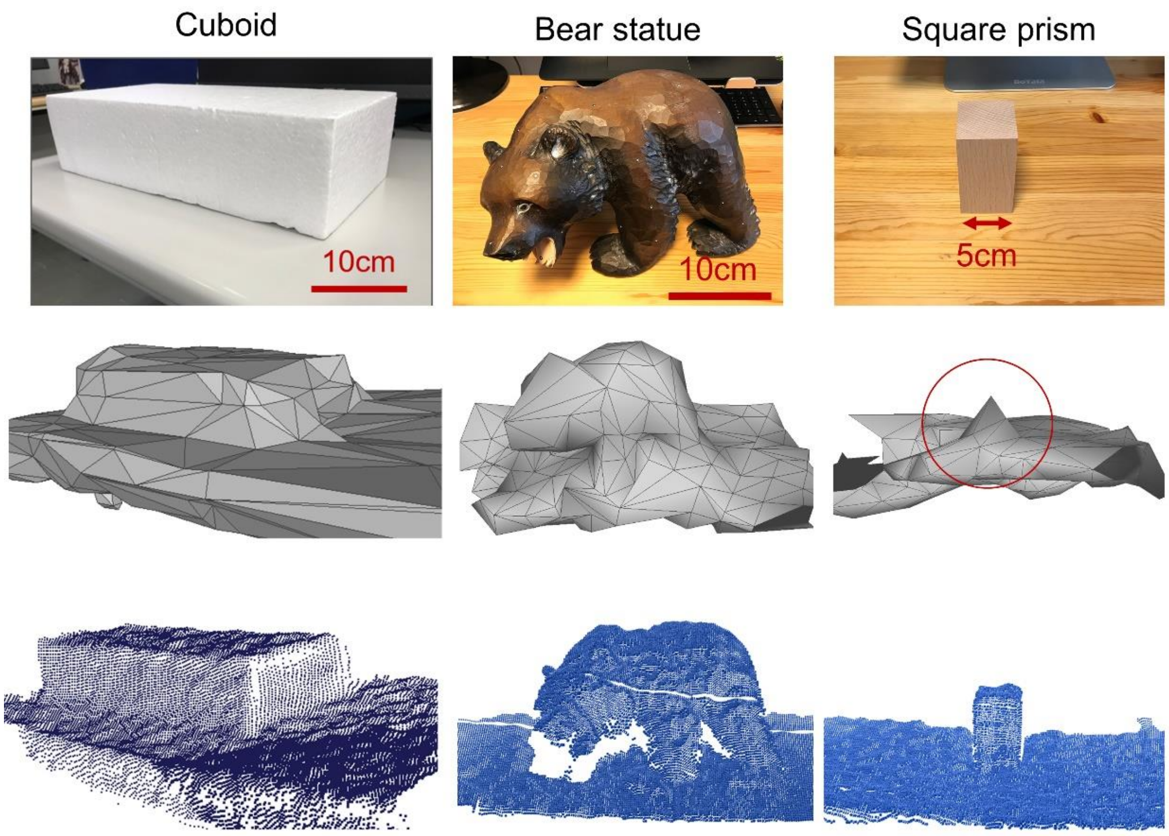


Fig. 2-11 Comparison of HoloLens-generated digital models with different objects. From top to bottom: scanned objects, mesh models, and point clouds.

In the case of a cuboid, although we can roughly identify its shape as a cuboid, its edges are not reconstructed well because of cm-level resolution of triangles. On the other hand, we can observe clear edges of the cuboid in its point cloud result which has precision in mm-level. Moreover, for the case of a bear statue that has more complex shape, we can hardly consider its mesh model as the same as the bear statue, but we can identify the clearer outline of the bear statue in the point cloud result.

The case of a square prism that is much smaller than the other two cases has a similar result. We cannot even identify its shape from its mesh model. Instead, it looks like a pyramid. Because the top surface of the square prism is smaller than the average size of the reconstructed triangle, the top surface is described as a point, and the whole shape becomes a pyramid-like shape.

Therefore, HoloLens-generated mesh models are better to be used to represent large object such as furniture that are larger than a cube with an edge of 30 cm.

On the other hand, with the point cloud information, we can use it to represent smaller objects. It shows that we can use point cloud data for the assembly composed of smaller real objects that are not presented well using HoloLens-generated models.

Next, we focus on the resolution of the point cloud along the depth direction, axis Z_C. The user wearing the HoloLens looked down at some objects as the top images in Fig. 2-12. A Preliminary test of scanning a flat table surface shows that the precision of z_C of the point cloud is approximately within ±4 mm. Thus, for a flat portable charger of height 10 mm that is put on a table, we can roughly identify its shape, but to have better scanning results, higher objects are preferred.

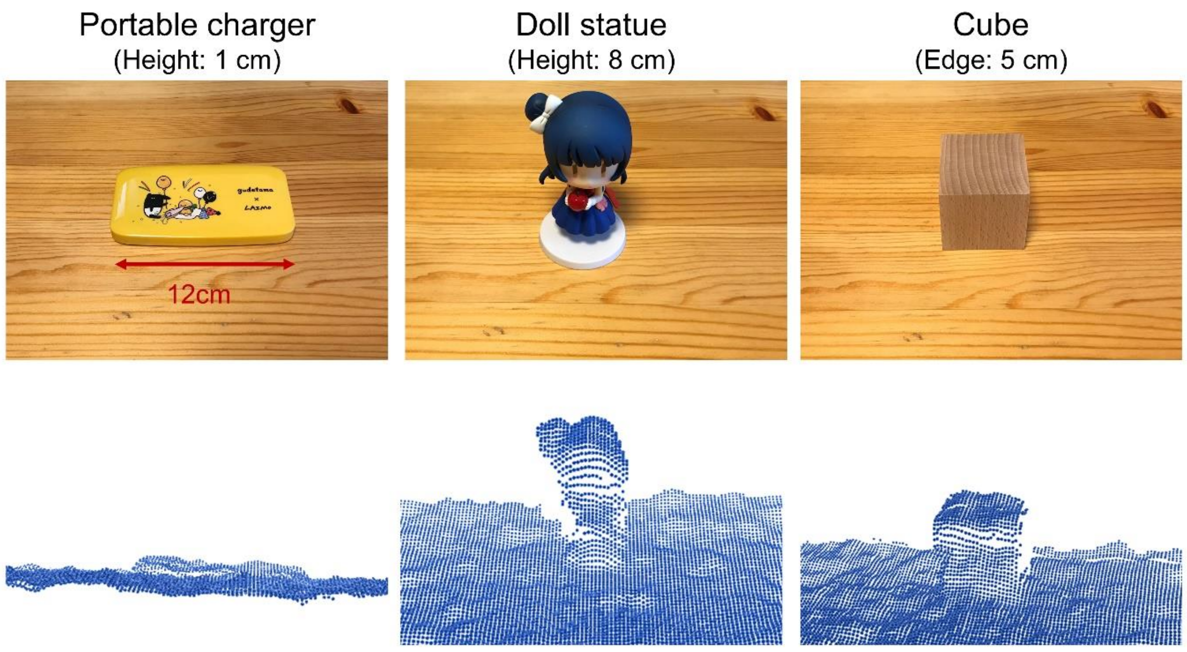


Fig. 2-12 Point cloud data of smaller objects whose height are lower than 8 cm

The case of a doll statue of height 8 cm shows that the depth camera can give only its outline. More detailed variation of the surfaces can not be preserved because the variations are lower than the precision of z_C. Moreover, for the case of a primitive cube with an edge of 5 cm, because of its primitive shape, its point cloud result can represent the object well. With these preliminary scanning results, using objects that have smooth surfaces and are larger than a cube with an edge of 2 cm are better target objects that can be represented well by the point cloud.

We next discuss the accuracy of the point cloud. We measure the position difference between the real object and the point cloud. In Fig. 2-13, by observing the point clouds in the coordinate system of the real world through the HoloLens, it is obvious that there is a shifting between the scanned surfaces of the real objects and the point clouds by a few centimeters. A few factors may cause this problem: (1) the bad scaling parameter to transform the measured flight time of IR light into the practical depth distance (a scaling problem), (2) user′s position in the real world is not well evaluated (a shifting problem), and (3) multipath interference [32][33][34] of IR light, which is one of common problems of time-of-flight depth cameras.

Multipath interference also influences the quality of the generated point cloud. It causes the distortion of the point cloud. In Fig. 2-13, we can find there is distortion in the right upper corner of the point cloud, which is not a right angle. To solve these problems, it is a way to calibrate the depth camera and then adjust the point cloud data [35][36][37]. However, there will a large number of factors to be considered, and the effect depends on different situations. Thus, instead, we handle this shifting problem by different methods in different stages of assembly operations. The detailed will be introduced in the following sections.

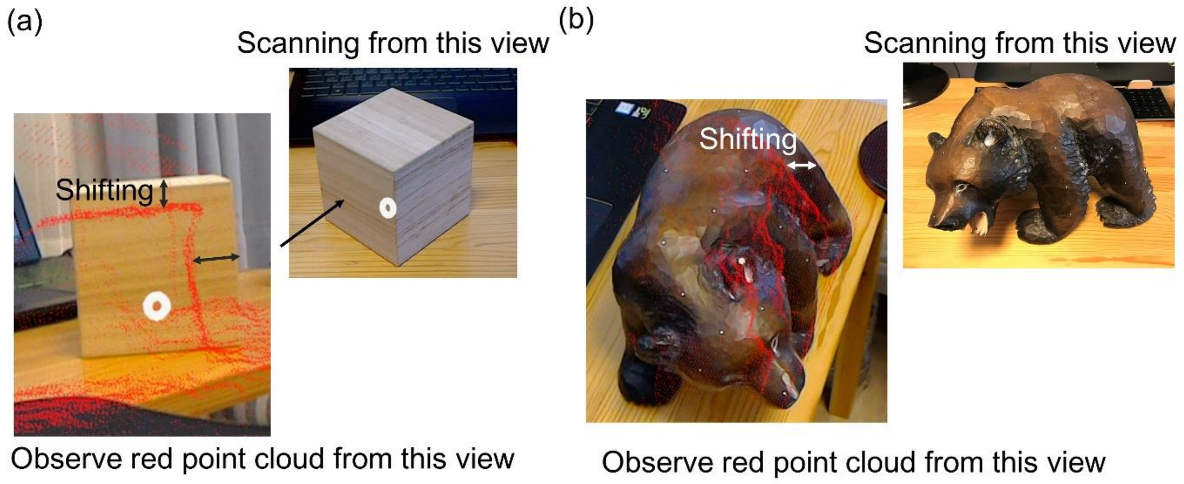

(a)
Scanning from this view
Shifting
Observe red point cloud from this view
(b)
Scanning from this view
Shifting
Observe red point cloud from this view

Fig. 2-13 Shifting of the generated point cloud in the real world: (a) a cube with an edge of 10 cm and (b) a bear statue

Finally, objects that are dark colors or transparent may not be scanned because they may absorb IR light, and no IR light will be reflected back to the depth camera. Thus, these objects can not be used in our application. We have to check real objects whether they can be scanned or not. In Fig. 2-14, we scan a wooden cube that is placed on different surfaces. The wooden cube is set as a reference to see scanning results of the surfaces. We can see that part of a wooden table around the cube is scanned. However, for a black plastic board, we can see that there is no point around the cube, and the return depth values from the depth camera is zero that means IR light is not reflect black. Although objects of dark color may not be scanned, not all objects of dark colors can not be scanned. For example, black clothes can be scanned in our test.

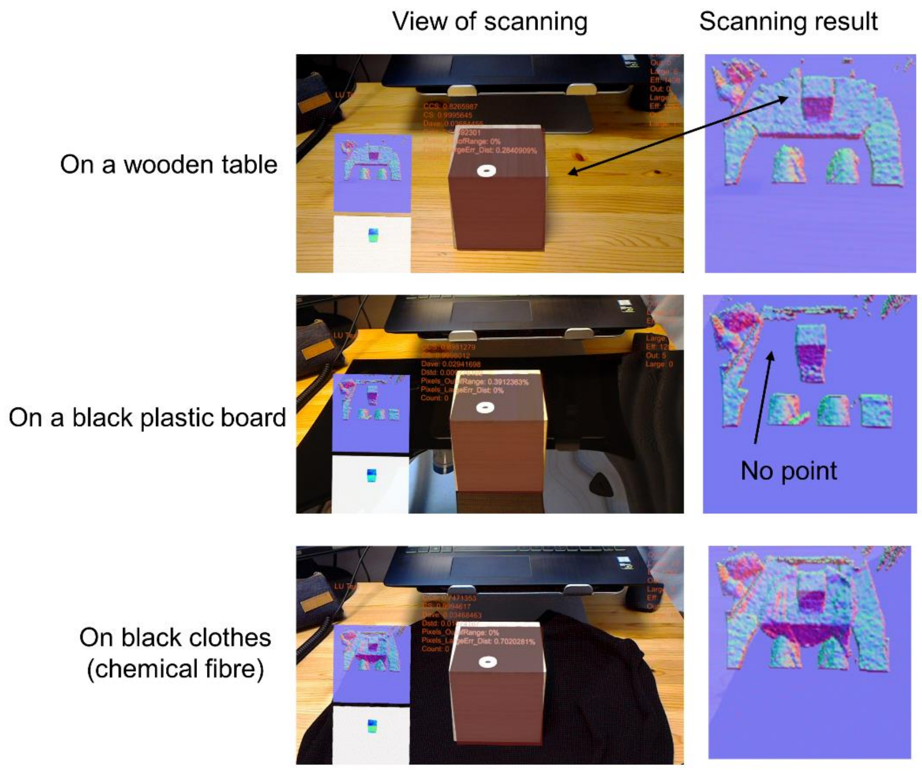


Fig. 2-14 Scanning results of objects of different materials and colors

2.4 Coordinate calibration between real and virtual world

Before starting assembly work, we need to perform coordinate calibration and adjustment between the real and virtual world. We need to define a working area, a table surface, to make the AR system display CAD models of the assembly and perform assembling operation on the table. For this purpose, we use a reference real object and its corresponding CAD model as a reference virtual object for coordinate calibration. In Fig. 2-15, a reference real object is placed on a table to define the working area which is around the reference real object. The CAD models of the assembly define the positions of parts in the assembly and their 3D models, and these CAD models are defined in the coordinate system of the virtual world in advance. In addition, we set a reference virtual object, the CAD model of the reference real object, at the origin of the virtual world. By transforming the reference virtual object to the position of the reference real object, the CAD models of the assembly will be transformed to the region of the working area together. Thus, the problem becomes how to determine the transformation relationship $\mathbf{T}_{RW \leftarrow VW}$, which denotes a matrix that transforms the origin of the virtual world to the designated position in the real world, the position of the reference real object.

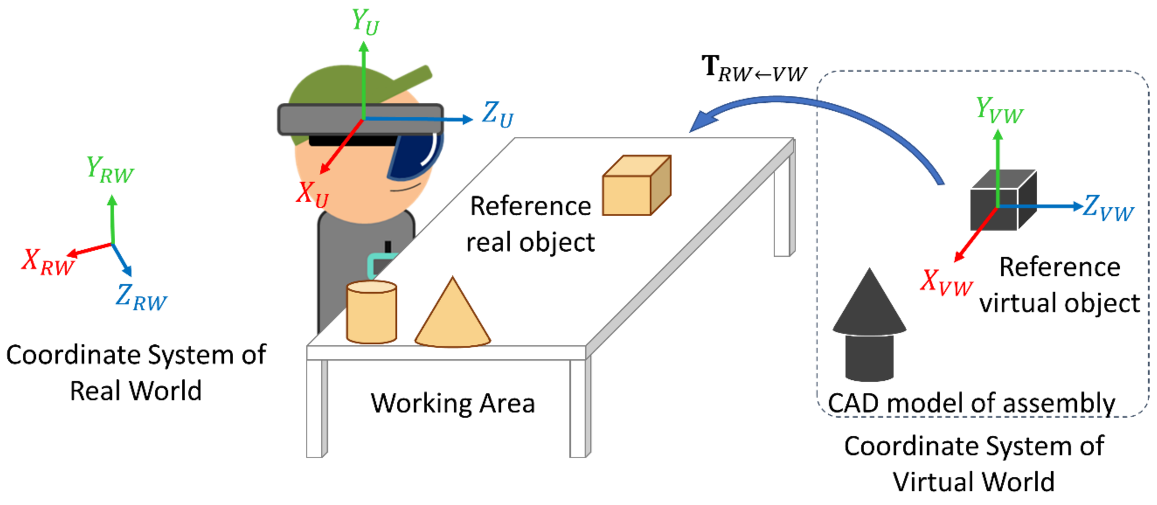


Fig. 2-15 Schematic plot of coordinate calibration

To determine the matrix $\mathbf{T}_{RW \leftarrow VW}$, we align the reference virtual object to the position of the reference real object with the corresponding orientation and then compute $\mathbf{T}_{RW \leftarrow VW}$. In Fig. 2-16, we design the alignment process involving two steps: rough alignment and precise alignment.

For rough alignment, we utilize a function called “Two Hand Manipulatable” in the Mixed Reality Toolkit and let the user move the reference virtual object to the reference real object roughly. The function uses the depth images in the near-depth sensing stream from the depth camera to track the user's hand manipulation as a controller to manipulate the target virtual object, as shown in Fig. 2-8(a). The translation control is based on the motion of two hands or one hand in the user's coordinate system Σ_U with X_U, Y_U, and Z_U axes. The hand movement in Z_U direction is identified by the depth information of tracked hands. The rotation control of the object is based on the relative motion of two hands. For instance, in Fig. 2-17, when the left hand moves backward, and the right hand moves forward, the virtual object will rotate counterclockwise about the Y_U axis. An illustration of rough alignment process is shown in Fig. 2-18.

Through hand manipulation, although the user can adjust the reference virtual object to the position of the reference real object, the user can only achieve the accuracy level of centimeter. The reason is that the HoloLens can not detect the hand movement in millimeter. Thus, we need to further apply another alignment to control the alignment error in millimeter level.

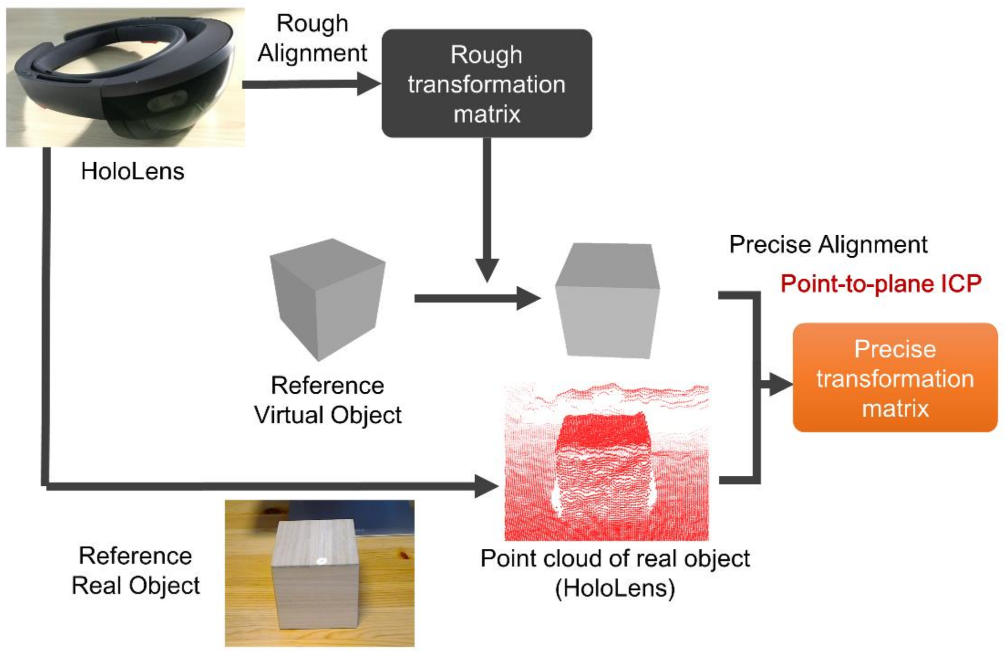


Fig. 2-16 Process of coordinate calibration

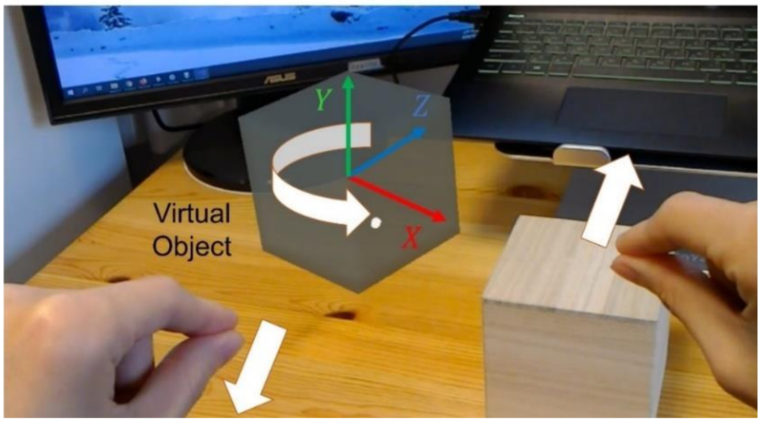


Fig. 2-17 Manipulate a virtual object through user's hands

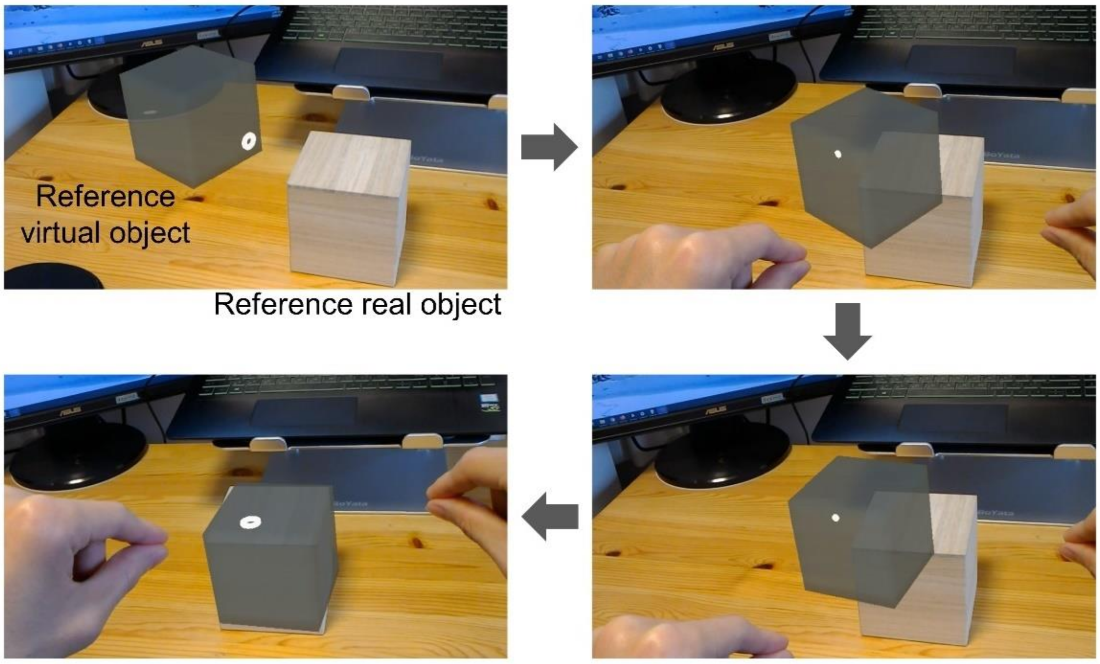


Fig. 2-18 Schematic plot of rough alignment process

After rough alignment, we perform precise alignment to make the reference virtual object overlap the real object in millimeter level. We use the point-to-plane iterative closest point (ICP) algorithm [38][39] to minimize the difference in positions of the two reference objects. To apply the point-to-plane ICP, we need their point cloud data. For the reference virtual object (i.e., a CAD model), we can acquire its point cloud data with the corresponding normal vector information for each point. For the digital content of the reference real object, we access the raw point cloud data from the depth camera as introduced in Section 2.3.2.

The process of precise alignment is shown in Fig. 2-19. First, before doing the point-to-plane ICP, a preprocessing is performed to extract and keep the scanned points of the real object inside the axis-aligned bounding box of the virtual object whose edges are aligned to the axes of the real-world coordinate system. This preprocessing removes most of the scanned points belonging to the surroundings that we are not interested and simultaneously decreases the number of points for finding pair-correspondences in the subsequent point-to-plane ICP. To keep the scanned points of the reference real object as many as possible, the size of the bounding box also plays an important role.

In Fig. 2-20, originally, the axis-aligned bounding box of the virtual object is defined by the smallest cuboid that fully encloses the virtual object. However, if the position of the virtual object after rough alignment is not close to the real object sufficiently, some scanned points of the reference real object will be removed in this stage. It will influence the final alignment result if there is a great loss of the scanned points of the reference real object. Therefore, to avoid this loss to some degree, we can simply expand the bounding box by increasing its size along each side. The remaining scanned points, which are inside the bounding box, mostly belong to the surfaces of the reference real object, and some of them are the points of the surroundings near the reference real object. With our preliminary tests, expanding the side of the bounding box by 2 cm is an appropriate choice to handle the rough alignment result with the accuracy of centimeters.

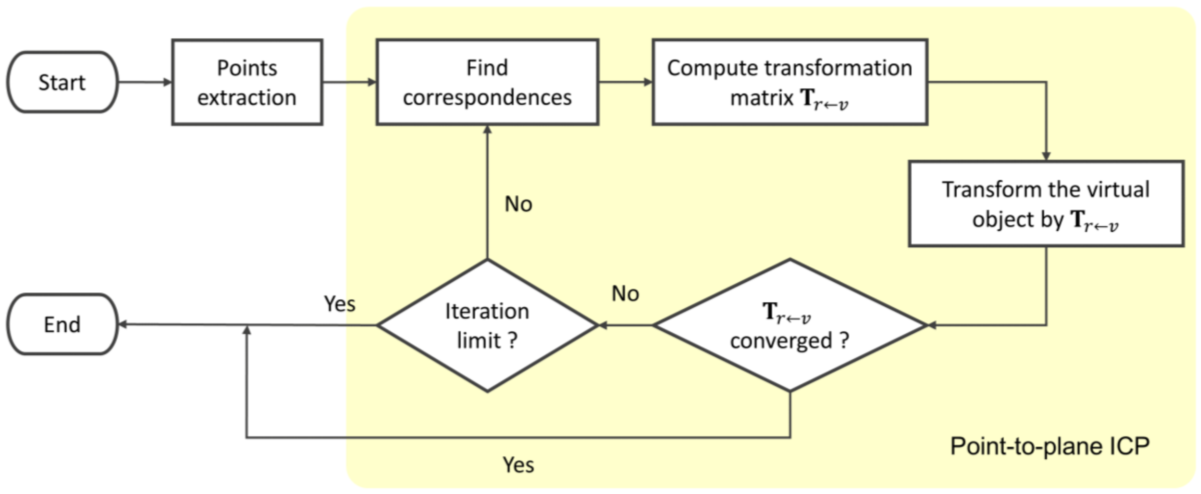

Start
Points extraction
Find correspondences
Compute transformation matrix $\mathbf{T}_{r\leftarrow v}$
Transform the virtual object by $\mathbf{T}_{r\leftarrow v}$
No
Iteration limit ?
No
$\mathbf{T}_{r\leftarrow v}$ converged ?
Yes
End
Yes
Point-to-plane ICP

Fig. 2-19 Process of precise alignment

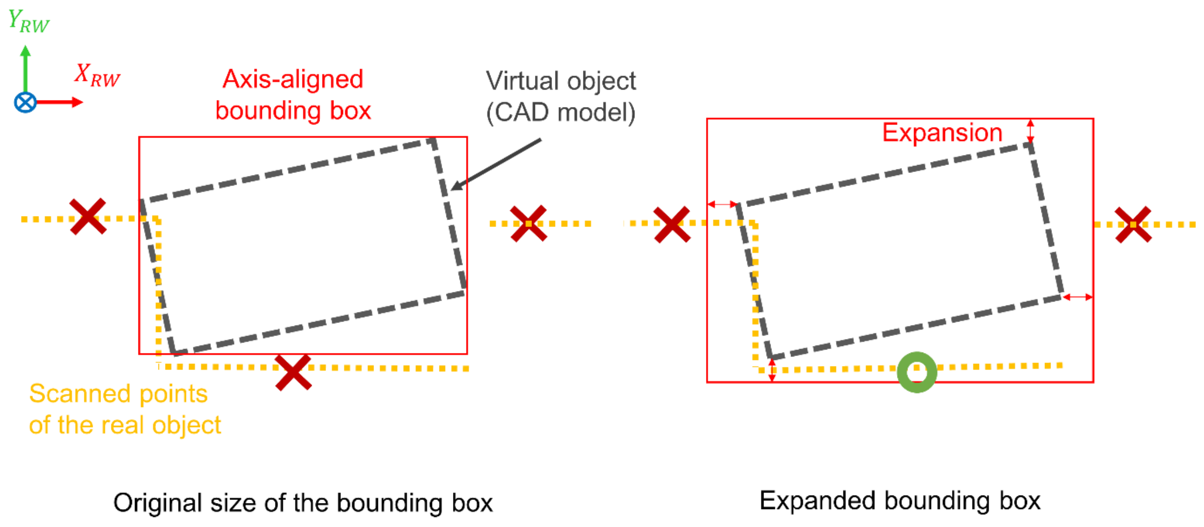

Y_{RW}
X_{RW}
Axis-aligned bounding box
Virtual object (CAD model)
Expansion
Scanned points of the real object
Original size of the bounding box
Expanded bounding box

Fig. 2-20 The illustration of the bounding box to extract the points that may belong to the reference real object.

Next, the point-to-plane ICP method is performed to achieve precise registration. Vertices of the virtual object (CAD model) will be registered to the HoloLens-scanned points of the real object. In each iteration of the point-to-plane ICP, it consists of three steps: (1) find correspondences, (2) compute a transformation matrix $\mathbf{M}_{r\leftarrow v}$ that minimizes an error function, and (3) transform the virtual object using evaluated $\mathbf{M}_{r\leftarrow v}$.

Find correspondences

A virtual object is defined as a triangular mesh with vertices $\{\mathbf{q}_i\}$. And $\{\mathbf{p}_i\}$ denotes the point cloud scanned by HoloLens. For each vertex $\mathbf{q}_i$ of the virtual object, its correspondence is the closest scanned point $\mathbf{p}_i$ associated with the minimum Euclidean distance, $\min\|\mathbf{p}_i - \mathbf{q}_i\|$. This is implemented by the k-d tree structure [40] to search for the closest scanned point fast. In addition, a threshold δ is set to further pick out reliable paired correspondences. Because the HoloLens-scanned points include only the partial information of the real object, it is preferred to take only potential corresponding points in the virtual object for the registration. Thus, if $\min\|\mathbf{p}_i - \mathbf{q}_i\| \leq \delta$, this paired correspondence $(\mathbf{p}_i, \mathbf{q}_i)$ will be selected into a group for the evaluation of the point-to-plane ICP; otherwise, $(\mathbf{p}_i, \mathbf{q}_i)$ will be excluded. Fig. 2-21 shows an illustration of finding effective correspondences.

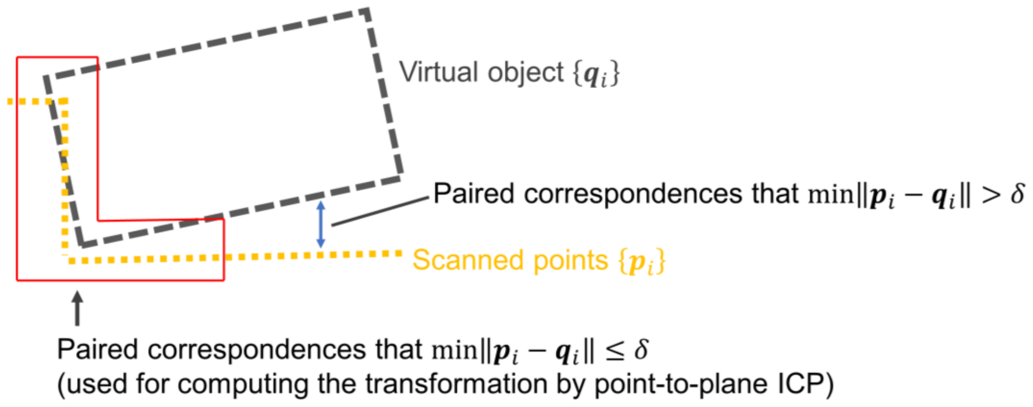

Virtual object $\{q_i\}$
Paired correspondences that $\min\|p_i - q_i\| > \delta$
Scanned points $\{p_i\}$
Paired correspondences that $\min\|p_i - q_i\| \leq \delta$
(used for computing the transformation by point-to-plane ICP)

Fig. 2-21 Paired correspondences searching based on the closest point

Compute the transformation matrix $\mathbf{M}_{r\leftarrow v}$ & Transform the virtual object

Based on all the paired correspondences, we next try to determine the transformation matrix $\mathbf{M}_{r\leftarrow v}$ such that

$$\mathbf{M}_{r\leftarrow v} = \underset{\mathbf{M}}{\operatorname{argmin}} \sum_{i=1}^{N} \left\|(\mathbf{M}\mathbf{q}_i - \mathbf{p}_i) \cdot \mathbf{n}_{\mathbf{q}_i}\right\|^2 \tag{6}$$

where N is the number of the paired correspondences found in the last step, and $\mathbf{n}_{\mathbf{q}_i}$ is the vertex normal vector associated with $\mathbf{q}_i$. $\mathbf{M}_{r\leftarrow v}$ and $\mathbf{M}$ are 3D transformations represented by 4 × 4 matrices using homogeneous coordinates, including rotation and translation:

$$\mathbf{M} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where $\{r_{ij}\}$ and $\{t_i\}$ are the parameters of rotation and translation, respectively. Moreover, $\mathbf{p}_i$, $\mathbf{q}_i$, and $\mathbf{n}_{\mathbf{q}_i}$ are also expressed in homogeneous coordinates. Eq. (6) suggests that the evaluated $\mathbf{M}_{r\leftarrow v}$ will make the dot product between vector $\overrightarrow{\mathbf{p}_i\mathbf{q}_i}$ and $\mathbf{n}_{\mathbf{q}_i}$ become close to zero. Geometrically, in Fig. 2-22, after the transformation of $\mathbf{M}_{r\leftarrow v}$, $\overrightarrow{\mathbf{p}_i\mathbf{q}_i}$ becomes $\overrightarrow{\mathbf{p}_i\mathbf{q}_i}'$ that is approximately orthogonal to $\mathbf{n}_{\mathbf{q}_i}$; that is, the scanned point $\mathbf{p}_i$ will be on the surface of the virtual object. Computation of $\mathbf{M}_{r\leftarrow v}$ is implemented according to Low's work [39] by linearizing Eq. (6) into a least-square optimization problem and then solving it by the singular value decomposition (SVD) method. After transforming the virtual object using the evaluated $\mathbf{M}_{r\leftarrow v}$, we perform the next iteration, repeating the above steps, until the upper limit of iterations or $\mathbf{M}_{r\leftarrow v}$ converges.

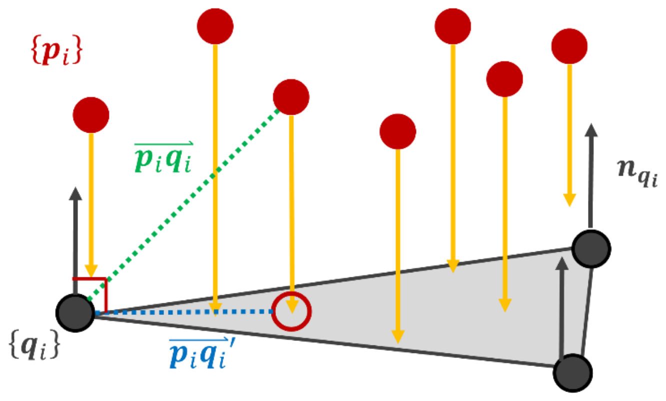

$\{\boldsymbol{p}_i\}$
$\overrightarrow{\boldsymbol{p}_i\boldsymbol{q}_i}$
$\boldsymbol{n}_{\boldsymbol{q}_i}$
$\{\boldsymbol{q}_i\}$
$\overrightarrow{\boldsymbol{p}_i\boldsymbol{q}_i}'$

Fig. 2-22 Schematic plot of the point-to-plane ICP

Although there is another ICP-related method called point-to-point ICP [41], which is also commonly used to minimize the difference between two point clouds, as shown in Fig. 2-23, and to reconstruct 2D or 3D surfaces from different views of scanning. Although each iteration of the point-to-plane ICP algorithm generally takes more time than the point-to-point ICP, Rusinkiewicz et al. [42] observed that the point-to-plane ICP has significantly better convergence rates. In our tests using Point Cloud Library (PCL) on a PC [43] to compare the two methods, the point-to-plane version generally takes less processing time. Thus, considering to the poorer computing ability of the HoloLens, we use the point-to-plane ICP method to perform precise alignment.

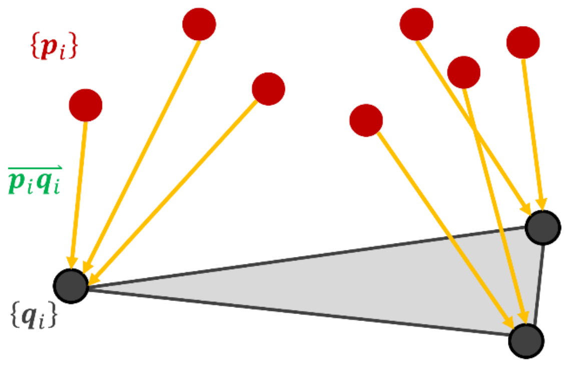


Fig. 2-23 Schematic plot of the point-to-point ICP

Performance test of precise alignment

To evaluate the performance of precise alignment, a wooden cube with an edge of 10 cm is used as the reference real object for the preliminary test. Before the performance evaluation, we first have to calibrate and adjust the HoloLens-scanned point cloud to make it attach to the surfaces of the real object. As described in Section 2.3.3, because of inappropriate calibration of the depth camera and interference from the environment, there seems a shifting between the real object and the scanned point cloud. This shifting may be caused by the inappropriate scaling parameter of the depth camera or the shifting of self-position in the real world. To mitigate this problem for precise alignment, we use the primitive cube and require the user to scan the cube from the specific view as shown in Fig. 2-24. The top and front planes of the cube are scanned, and the next is to adjust the scanned point cloud to make it attached to the surfaces of the cube.

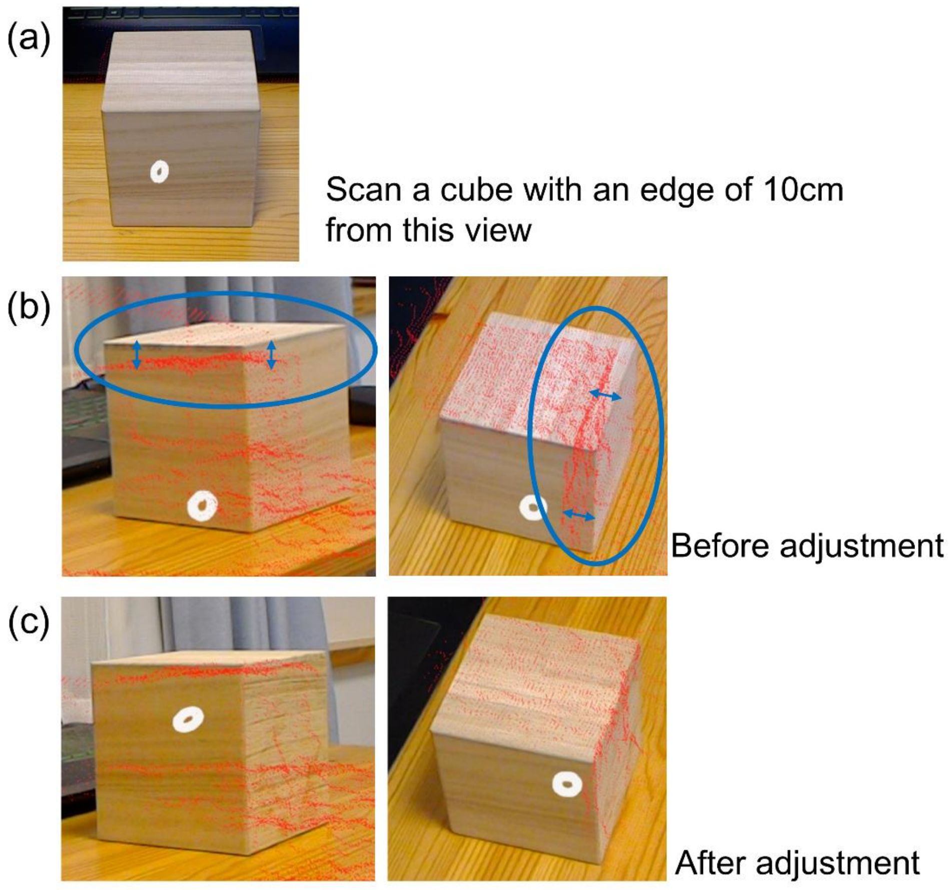


Fig. 2-24 Adjust the point cloud for coordinate calibration

Moreover, we find that by scanning the cube at a close distance, approximately shorter than 30 cm, it results in approximately constant shift amounts of 1 cm vertically (top surface) and of 1.6 cm horizontally along axis Z_U (front surface). Otherwise, larger distances of scanning generally lead to larger shifting amounts by a few centimeters. This may derive from multipath interference that larger distances cause larger errors. By scanning the cube from the specific view and at the close distance, the shift amount for the adjustment of the point cloud may be repeatedly used in future tasks.

After adjusting the point cloud of the reference real object and rough alignment, we can perform precise alignment. In Fig. 2-25, a corresponding CAD model of the cube is constructed with 386 vertices, 768 faces, and normal vectors for all vertices. Fig. 2-26(a) shows the point cloud of the real scene, a cube on a table, and Fig. 2-26(b) shows the position of the virtual object after rough alignment, which is not well aligned to the scanned points of the real object. After rough alignment, we can start to perform precise alignment. First, point extraction was performed to remove redundant scanned points outside the bounding box of the virtual object. Here, we increased the extent of the bounding box by 2 cm to keep potential points belonging to the real object. As a result, in Fig. 2-26(c), the number of the scanned points decreased from 65,536 to 5,252 points.

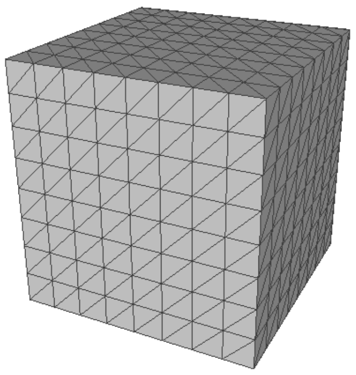


Fig. 2-25 The CAD model of the reference object

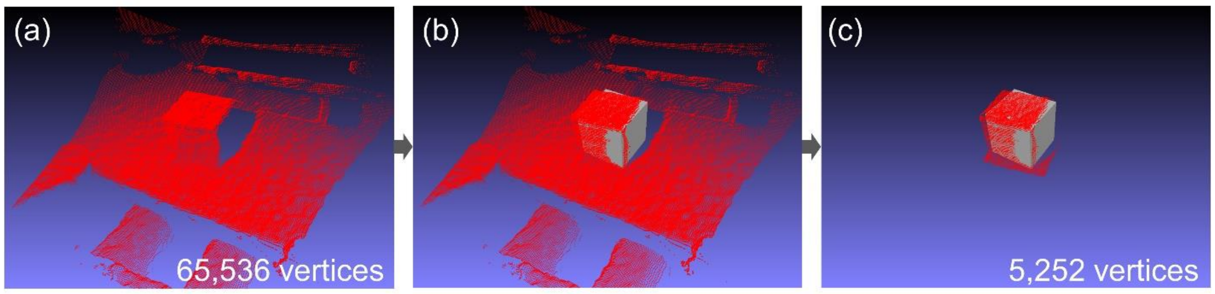


Fig. 2-26 Extract the scanned points near the virtual object: (a) scanned scene, (b) scanned scene with the virtual object after rough alignment, and (c) scanned points inside the bounding box of the virtual object.

Afterward, most points belonging to the top and front surfaces of the real object were kept. Those preserved 5,252 scanned points were used to perform point-to-plane ICP with the virtual object. Fig. 2-27 shows the experimental results with different iterations. In our preliminary development, it took 1.5 sec for 10 iterations of the point-to-plane ICP on the HoloLens. Basically, after 5th iteration, the transformation of the virtual object converged and was not obvious in this case. It can be seen that the virtual object (mesh of the cube) eventually attached well to the scanned points of the real cube (red points). We can see that the position and orientation of the virtual object also became much closer to the real one in the mixed reality photos (user's view).

Thus, starting from rough alignment to precise alignment, a series of transformations forms the transformation matrix $\mathbf{T}_{RW \leftarrow VW}$ that transforms the virtual object from the original position, somewhere predefined in the real-world coordinate system, to the position of the corresponding real object, which can be defined by the user arbitrarily.

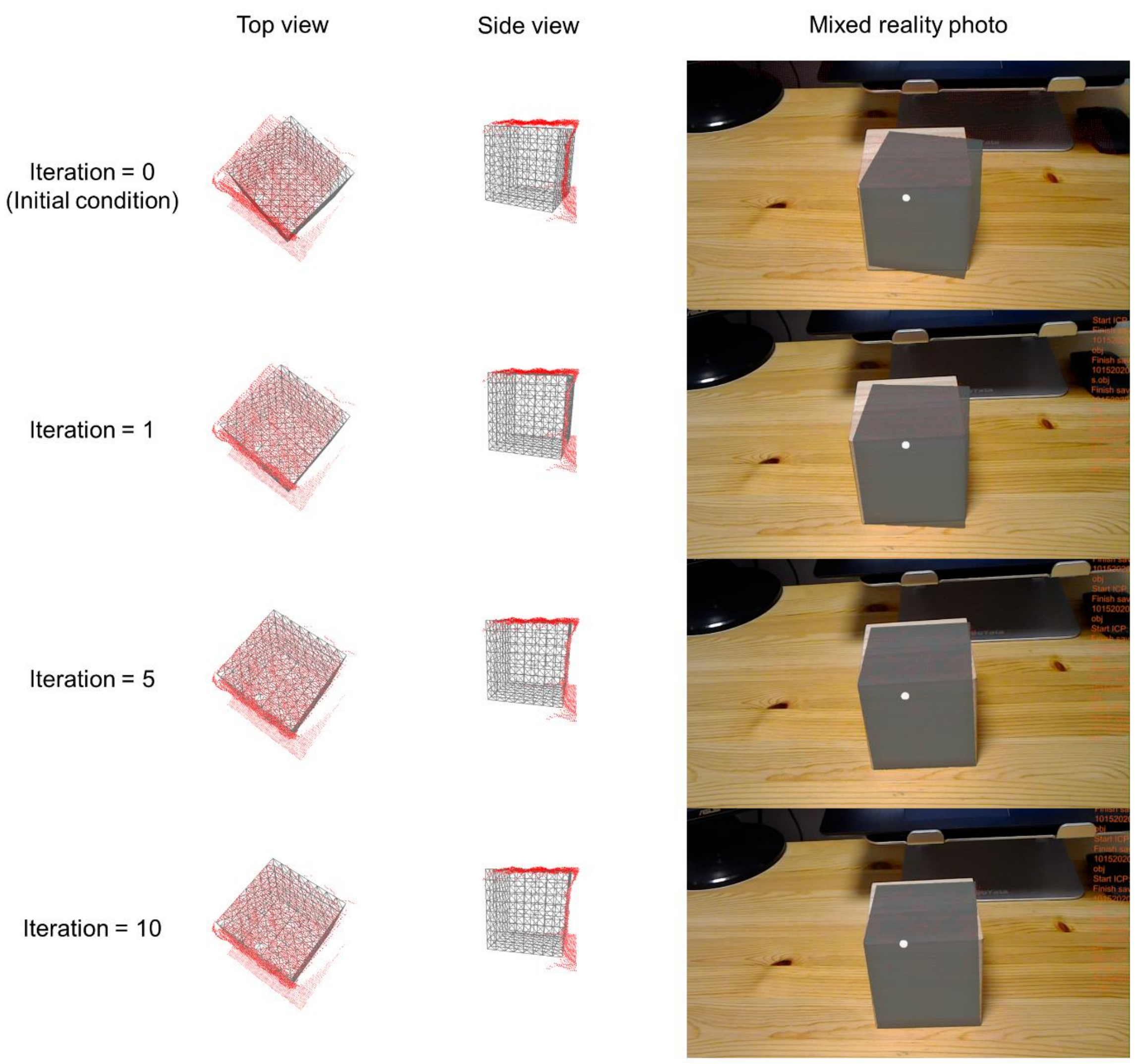


Fig. 2-27 Registration results of the point-to-plane ICP

2.5 Evaluation of misalignment between real and virtual objects

2.5.1 Design of misalignment evaluation

After coordinate calibration, the system can start to display the virtual objects (CAD models) of the assembly in the preferred working area. During the assembling operation, a user holds the corresponding real object and tries to move it to the position of the virtual object. In order to assist the user to place the real object to the right position, the system evaluates whether there is misalignment between the real and virtual objects to check whether the assembling operation is correct. Because we can acquire the point cloud data of real objects from the HoloLens and have the corresponding CAD models, misalignment evaluation between the real and virtual objects can be realized by comparing the distance or similarity between the scanned point cloud of the real object and the CAD model, the virtual object as assembling indication.

The most general approach for evaluating the misalignment between two objects in the 3D space is to compute distance between them. This computation requires computation of finding corresponding points between the CAD model and the point cloud of the real object and calculating their distances as well. In our application, this misalignment evaluation must be done at real time rate in order to indicate the misalignment immediately following the object handling motion of the user. In this aspect, this approach is too expensive for the HoloLens with small computation resources. It is crucial to take more efficient approach for misalignment evaluation. Instead, we compare the depth images of the real and virtual objects to evaluate misalignment, i.e., performing evaluation in 2D space.

Fig. 2-28 shows the process of obtaining depth images for the case of a cube on a table. We set two virtual depth cameras that has the same position and orientation as the physical one in the HoloLens. We can then access the depth maps of the scanned point cloud and of the CAD model that are rendered by the GPU in the HoloLens from the view of the virtual cameras. Related implementation is introduced in Sec. 2.6.1. Because the two depth maps are generated from the same viewpoint, if now the real object and the virtual object are matched, we can expect that the two depth maps will be similar.

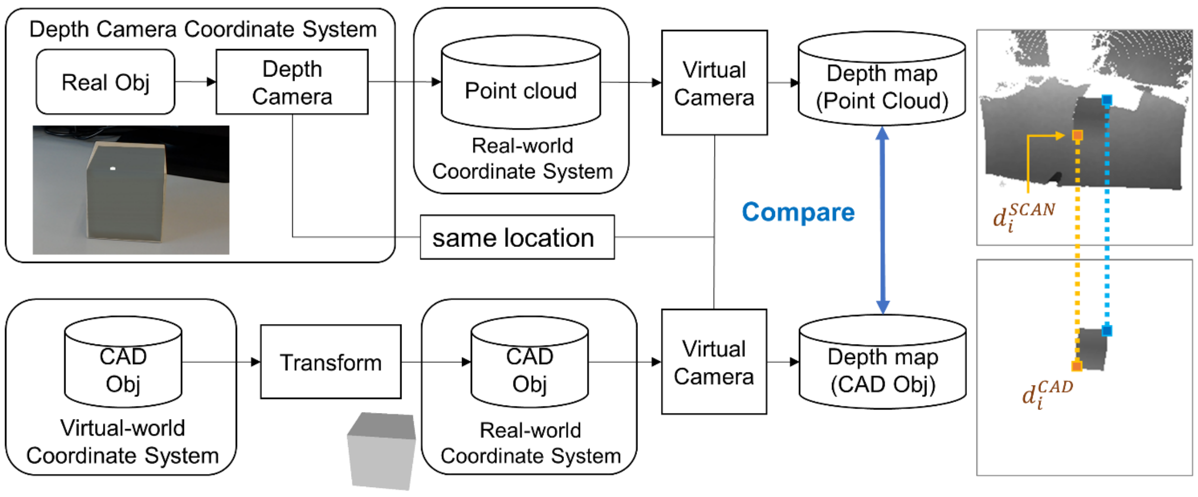


Fig. 2-28 Process of obtaining GPU-rendered depth images of the scanned point cloud and the CAD model.

Notice that we access the GPU-generated depth map of the scanned point cloud and use it to represent the depth map of the real object instead of directly using the depth image from the physical depth camera. Because the depth map of the CAD model is generated by the GPU, using the GPU-generated depth map of the scanned point cloud can make the two depth maps in the same pixel coordinate system for the convenience of the subsequent misalignment evaluation. Moreover, the original depth image from the physical depth camera has the effect of lens distortion, but the depth map of the CAD model does not have. Thus, we first transform the original depth image to the point cloud, which is generated with the correction of lens distortion as introduced in Section 2.3.2, and then access the GPU-generated depth map of the point cloud.

2.5.2 Analysis of different evaluation methods

After obtaining the depth maps of the scanned point cloud and the CAD model in the same coordinate system, we can proceed to compare these two depth maps to evaluate the misalignment of the real and virtual objects. We define several functions of indicators for such misalignment error. From the depth maps of the CAD model, we know which pixels belong to the CAD model. If the real and virtual objects are matched, it can be expected that the depth values in the same pixel region will be similar, and we can compare those pixels in the two depth maps for misalignment alignment, as shown in Fig. 2-29(a).

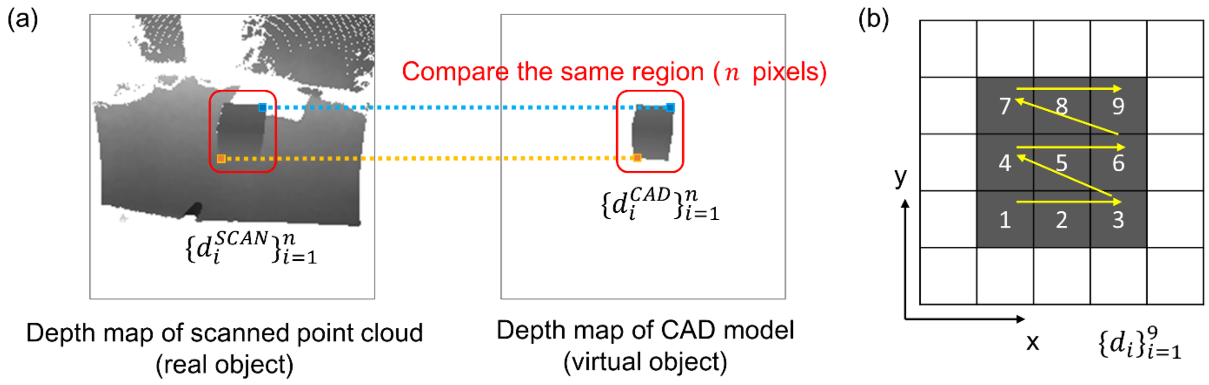

(a)
Compare the same region (n pixels)
$\{d_i^{SCAN}\}_{i=1}^{n}$
$\{d_i^{CAD}\}_{i=1}^{n}$
Depth map of scanned point cloud (real object)
Depth map of CAD model (virtual object)
(b)
7 8 9
4 5 6
1 2 3
y
x
$\{d_i\}_{i=1}^{9}$

Fig. 2-29 Schematic plot of comparing the depth maps: (a) compare the same region in the depth maps and (b) rearrange pixels into a sequence in the raster scanning order

For comparison, we rearrange the depth values in the target pixel region into a sequence in bottom-to-top and left-to-right order, as shown in Fig. 2-29(b). Therefore, we obtain the sequences of the depth values of $\{d_i^{SCAN}\}_{i=1}^{n}$ and $\{d_i^{CAD}\}_{i=1}^{n}$ for the depth maps of the scanned point cloud and of the CAD model, respectively where n is the number of pixels that belong to the CAD model in the depth map of the CAD model. The problem of misalignment evaluation is then converted into measuring the similarity between $\{d_i^{SCAN}\}_{i=1}^{n}$ and $\{d_i^{CAD}\}_{i=1}^{n}$.

To compare the sequences, we study some methods to determine an effective one as an indicator to evaluate whether there is misalignment between the real and virtual object:

The average and standard deviation of the depth differences

We first simply compute the arithmetic mean and standard deviation of the depth differences, D_{ave} and D_{std}, as two potential indicators:

$$D_{ave} = \frac{\sum_{i=1}^{n} \Delta d_i}{n},$$

and

$$D_{std} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(\Delta d_i - D_{ave})^2}$$

where $\Delta d_i = \left|d_i^{SCAN} - d_i^{CAD}\right|$. These indicators evaluate misalignment based on the distance, the difference of the depth values, between the scanned point cloud and the CAD model. When the real and virtual objects are matched, it can be expected that the objects are close, so D_{ave} and D_{std} will be both small; otherwise, it demonstrates that misalignment occurs.

Cosine similarity and centered cosine similarity

Next, we study the cosine similarity and centered cosine similarity methods as another potential indicators. These methods measure the similarity between two non-zero vectors according to the cosine of the angle between them, which is the same as the dot product of the two vectors normalized to have length of 1. Suppose $\{p_i\}_{i=1}^{n}$ and $\{q_i\}_{i=1}^{n}$ are sequences of real numbers and define two different data. Let **p** and **q** be ordered *n*-element vectors containing the above data:

$$\mathbf{p} = (p_1, p_2, \dots, p_n) \text{ and } \mathbf{q} = (q_1, q_2, \dots, q_n)$$

in $\mathbb{R}^n$. The cosine similarity between **p** and **q** is defined by

$$CS(\mathbf{p}, \mathbf{q}) = \frac{\mathbf{p} \cdot \mathbf{q}}{\|\mathbf{p}\|\|\mathbf{q}\|} = \frac{\sum_{i=1}^{n} p_i q_i}{\sqrt{\sum_{i=1}^{n} {p_i}^2}\sqrt{\sum_{i=1}^{n} {q_i}^2}} \tag{7}$$

A small angle between two vectors, which gives a high result of the cosine similarity, indicates high similarity between the vectors while a large angle, low cosine similarity, means low similarity. In addition, the output range of the cosine similarity is [-1, 1] for any included angle between two vectors in the interval [0°, 180°]. Two vectors that have the same direction have the included angle of 0°, and its cosine similarity is 1. Two vectors that are orthogonal have the angle of 90° and the cosine similarity of 0, and two vectors in the opposite directions have the angle of 180° and the cosine similarity of -1. Fig. 2-30 shows the illustration of the cosine similarity result, which measures the cosine of the angle instead of the distance between the two vectors.

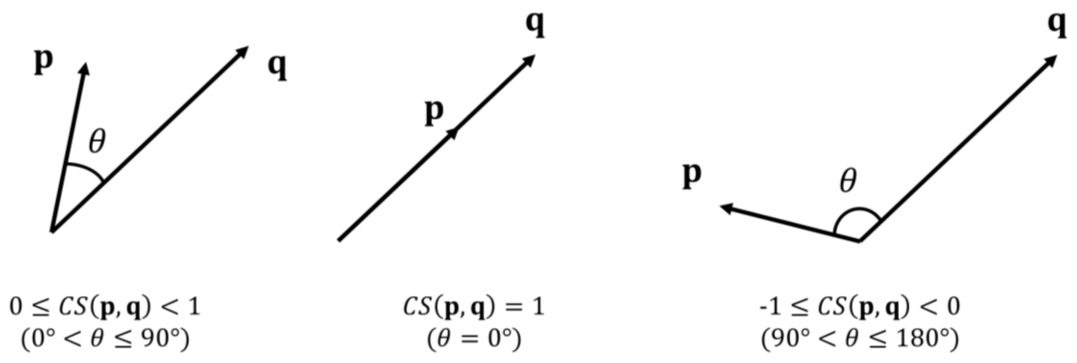

p
q
θ
$0 \le CS(\mathbf{p},\mathbf{q}) < 1$
$(0° < \theta \le 90°)$
p
q
$CS(\mathbf{p},\mathbf{q}) = 1$
$(\theta = 0°)$
p
q
θ
$-1 \le CS(\mathbf{p},\mathbf{q}) < 0$
$(90° < \theta \le 180°)$

Fig. 2-30 Schematic plot of the cosine similarity result

Moreover, the centered cosine similarity is defined by

$$CCS(\mathbf{p},\mathbf{q}) = \frac{\sum_{i=1}^{n}(p_i - \bar{p})(q_i - \bar{q})}{\sqrt{\sum_{i=1}^{n}(p_i - \bar{p})^2}\sqrt{\sum_{i=1}^{n}(q_i - \bar{q})^2}} \tag{8}$$

where $\bar{p}$ and $\bar{q}$ are the arithmetic mean of $\{p_i\}_{i=1}^{n}$ and $\{q_i\}_{i=1}^{n}$, respectively. The output range is the same as the output of the cosine similarity, [-1, 1], for any included angle between two vectors in the interval [0°, 180°]. Eq. (8) shows that the centered cosine similarity is the cosine similarity between centered versions of $\mathbf{p}$ and $\mathbf{q}$.

In particular, the cosine similarity is invariant to the scaling of data. On the other hand, the centered cosine similarity is invariant to the shift and scaling of data. About these features, we first prove the shift and scale invariance of the centered cosine similarity. For any constants a, b, c, and d with a, $c > 0$, we may scale and shift the data $\{p_i\}_{i=1}^{n}$ and $\{q_i\}_{i=1}^{n}$, and then generate new data $\{p_i'\}_{i=1}^{n}$ and $\{q_i'\}_{i=1}^{n}$ where $p_i' = ap_i + b$ and $q_i' = cq_i + d$. Let $\mathbf{p}' = (p_1', p_2', \ldots, p_n')$ and $\mathbf{q}' = (q_1', q_2', \ldots, q_n')$ be n-dimensional vectors from $\{p_i'\}_{i=1}^{n}$ and $\{q_i'\}_{i=1}^{n}$, respectively. The centered cosine similarity between $\mathbf{p}'$ and $\mathbf{q}'$ is then given by

$$CCS(\mathbf{p}',\mathbf{q}') = \frac{\sum_{i=1}^{n}(p_i' - \bar{p}')(q_i' - \bar{q}')}{\sqrt{\sum_{i=1}^{n}(p_i' - \bar{p}')^2}\sqrt{\sum_{i=1}^{n}(q_i' - \bar{q}')^2}}$$

where $\bar{p}'$ and $\bar{q}'$ are the arithmetic mean of $\{p_i'\}_{i=1}^{n}$ and $\{q_i'\}_{i=1}^{n}$, respectively.

Particularly,

$$\bar{p}' = a\bar{p} + b,$$

and

$$\bar{q}' = c\bar{q} + d.$$

Consequently,

$$\begin{aligned} CCS(\mathbf{p}', \mathbf{q}') &= \frac{\sum_{i=1}^{n}(ap_i + b - a\bar{p} - b)(cq_i + d - c\bar{q} - d)}{\sqrt{\sum_{i=1}^{n}(ap_i + b - a\bar{p} - b)^2}\sqrt{\sum_{i=1}^{n}(cq_i + d - c\bar{q} - d)^2}} \\ &= \frac{ac\sum_{i=1}^{n}(p_i - \bar{p})(q_i - \bar{q})}{a\sqrt{\sum_{i=1}^{n}(p_i - \bar{p})^2} \times c\sqrt{\sum_{i=1}^{n}(q_i - \bar{q})^2}} \\ &= \frac{\sum_{i=1}^{n}(p_i - \bar{p})(q_i - \bar{q})}{\sqrt{\sum_{i=1}^{n}(p_i - \bar{p})^2}\sqrt{\sum_{i=1}^{n}(q_i - \bar{q})^2}} = CCS(\mathbf{p}, \mathbf{q}). \end{aligned}$$

This implies that the result of the centered cosine similarity is invariant to the shift and scaling of data. The scale invariance of the cosine similarity can be proved similarly by substituting the shifts b and d into zero.

Moreover, the cosine similarity and the centered cosine similarity measure whether the two sequences have the same variation trend; that is, if larger p_i (smaller p_i) corresponds to larger q_i (smaller q_i), the results will be high (low). Thus, it is a possible way to use the cosine similarity and the centered cosine similarity as indicators to measure the similarity between $\{d_i^{SCAN}\}_{i=1}^{n}$ and $\{d_i^{CAD}\}_{i=1}^{n}$, i.e.,

$$D_{CS} = CS(\mathbf{d}^{SCAN}, \mathbf{d}^{CAD}) \text{ and } D_{CCS} = CCS(\mathbf{d}^{SCAN}, \mathbf{d}^{CAD})$$

where $\mathbf{d}^{SCAN} = (d_1^{SCAN}, d_2^{SCAN}, \dots, d_n^{SCAN})$ and $\mathbf{d}^{CAD} = (d_1^{CAD}, d_2^{CAD}, \dots, d_n^{CAD})$, respectively. D_{cs} and D_{ccs} measure whether the depth maps of the scanned point cloud and the CAD model have the similar variation trend of the depth values instead of measuring the difference of the depth values as D_{ave} and D_{std}.

2.5.3 Performance test of different evaluation methods

As introduced in Section 2.3.2, because of the errors of the HoloLens-scanned point cloud, we need a robust indicator to evaluate the misalignment effectively. With the above-mentioned evaluation methods, the next step is to design experiments to characterize the performance of the different evaluation methods and to determine an effective one as the indicator for misalignment evaluation. The experimental setup is shown in Fig. 2-31(a). We put a real object and the corresponding virtual object on a table surface that is parallel to $X_{RW}Z_{RW}$ plane and evaluate misalignment between the real and virtual objects using the four indicators, D_{ave}, D_{std}, D_{cs}, and D_{ccs}.

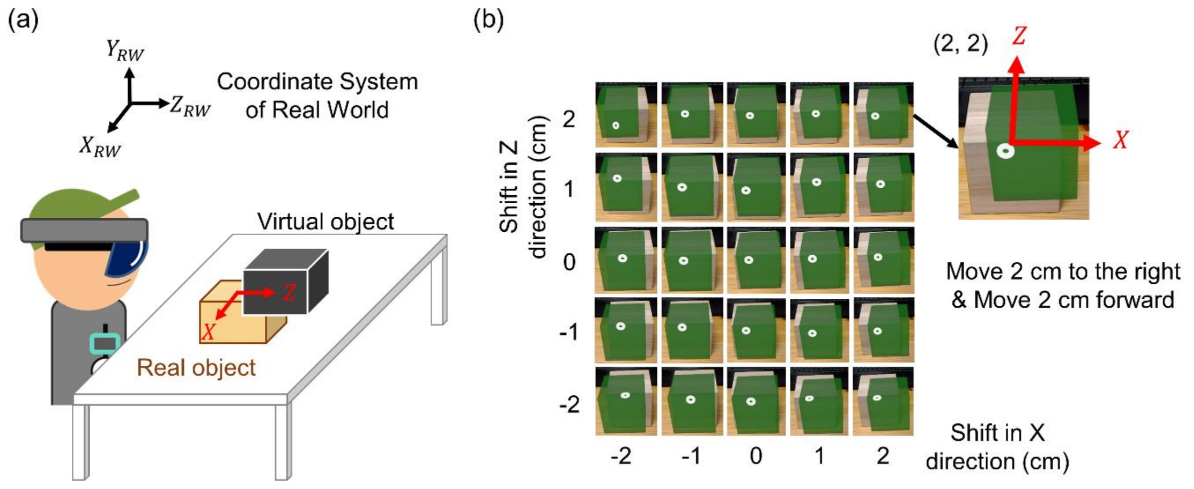

(a)
Y_{RW}
Z_{RW}
X_{RW}
Coordinate System of Real World
Virtual object
Z
X
Real object
(b)
Shift in Z direction (cm)
2
1
0
-1
-2
-2
-1
0
1
2
(2, 2)
Z
X
Move 2 cm to the right & Move 2 cm forward
Shift in X direction (cm)

Fig. 2-31 Experimental setup for the performance tests of the indicators (a) fixed real object and the associated movable virtual object and (b) design of shifting the virtual object to different positions

To do evaluation under different misalignment conditions, we shift the virtual object to different positions on the table (*XZ* plane). In Fig. 2-31(b), by setting the origin of the *XZ* coordinate system at the center of the real object, the virtual object moves relative to the real object along the *X* and *Z* axes. For example, at (0, 0), it means the real and virtual objects are matched, and at (2, 2), it implies the virtual object moves 2 cm to the right and 2 cm forward relative to the real object. With this experimental design, we investigate four objects with different shapes and sizes, as shown in Fig. 2-32: (1) a cube with an edge of 10 cm, (2) a bear statue, (3) a cube that is the same as (1) but rotates by 45°, and (4) a triangular block that is the half size of a cube with an edge of 5 cm.

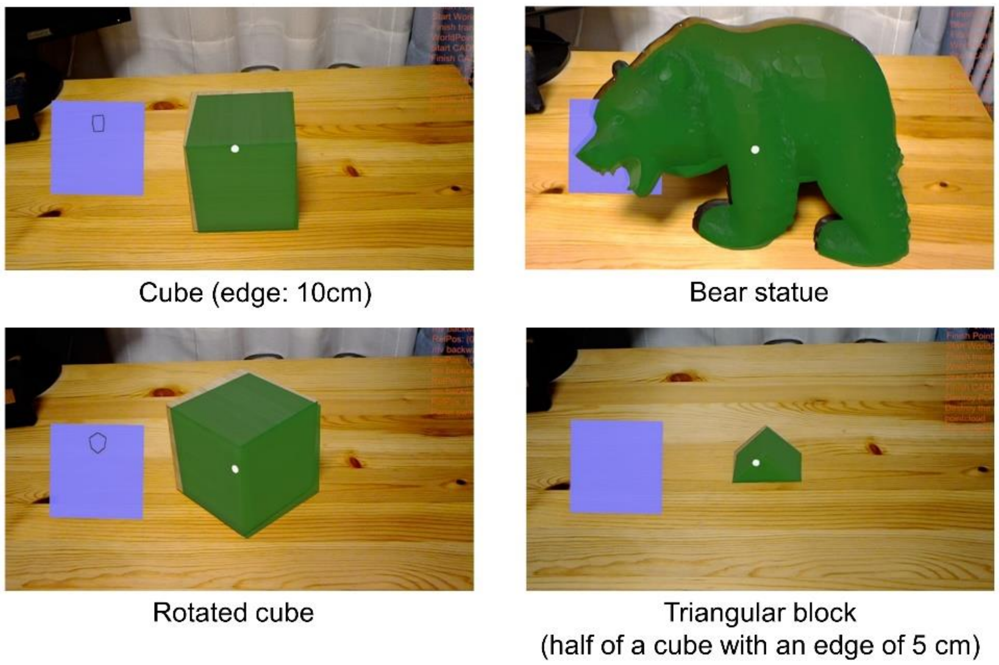
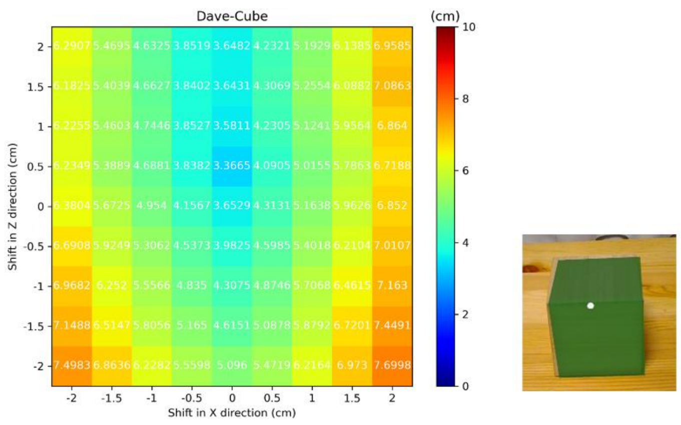


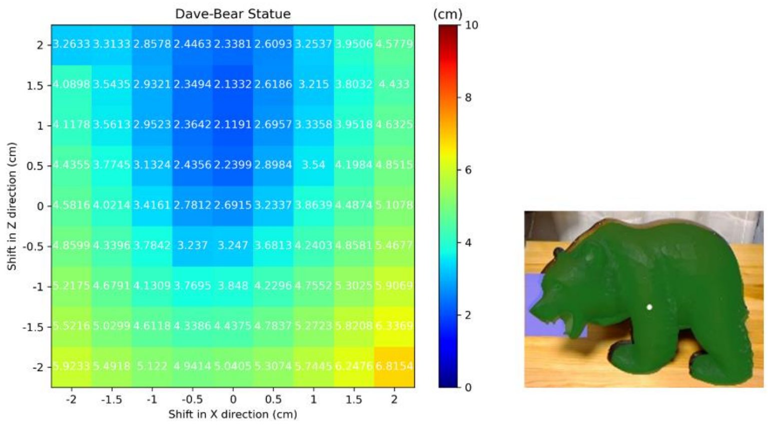
Fig. 2-32 Objects with different shapes and sizes to test the indicator performance

Fig. 2-33 and Fig. 2-34 show experimental results using the D_{ave}, D_{std} indicators, respectively. It can be seen that when the positions of the real and virtual objects are closer, both the D_{ave} and D_{std} indicators give smaller values. It shows that the difference of the depth values between $\{d_i^{SCAN}\}_{i=1}^{n}$ and $\{d_i^{CAD}\}_{i=1}^{n}$ are smaller, and the two sequences are more similar. However, for the case of the triangular block, the D_{ave} and D_{std} values do not change apparently with the change of the virtual object′s position in comparison with the cases of the other three objects. For example, in Fig. 2-33(b) and Fig. 2-34(b), if we move the virtual bear statue from position $(-2, 0)$ to $(0, 0)$, the D_{ave} and D_{std} values decrease by 1.89 cm and 4.665 cm, respectively. On the other hand, in Fig. 2-33(d) and Fig. 2-34(d), with the same shifts of the virtual triangular block, the D_{ave} and D_{std} values decrease by only 0.012 cm and 0.748 cm, respectively.

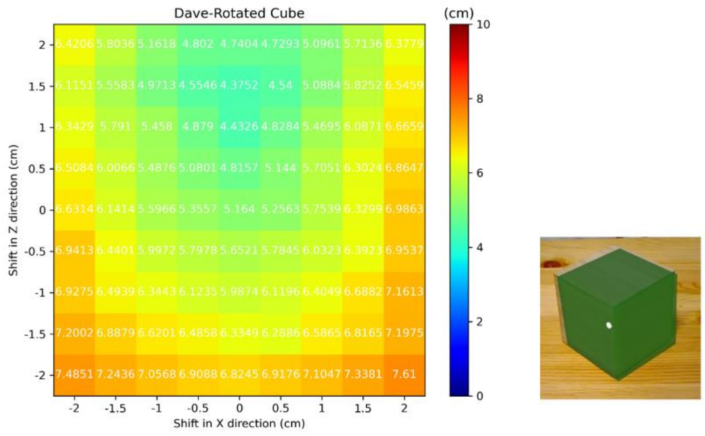
(a)



(b)



(c)



(d)

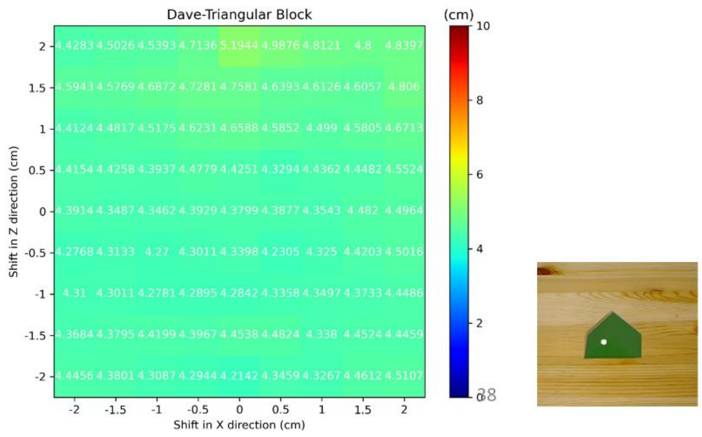
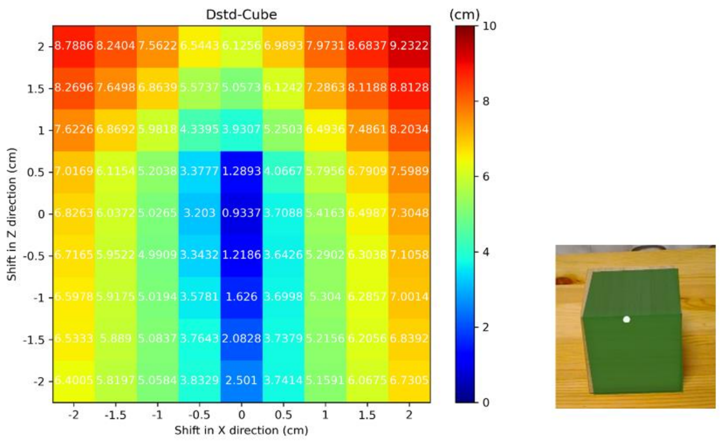
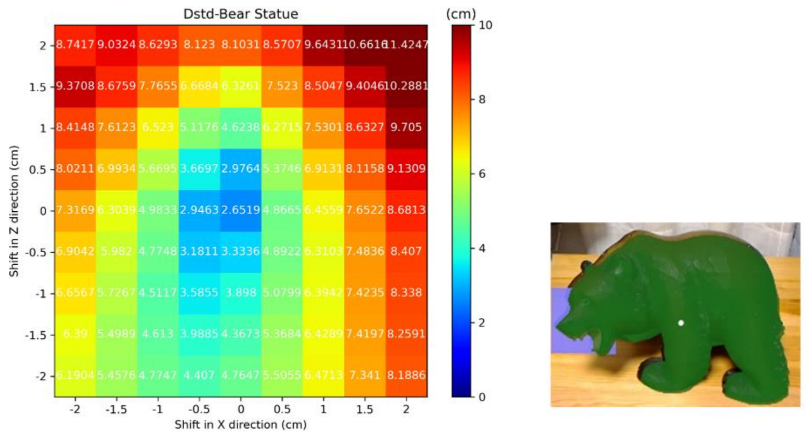


Fig. 2-33 Evaluation results using the D_{ave} indicator

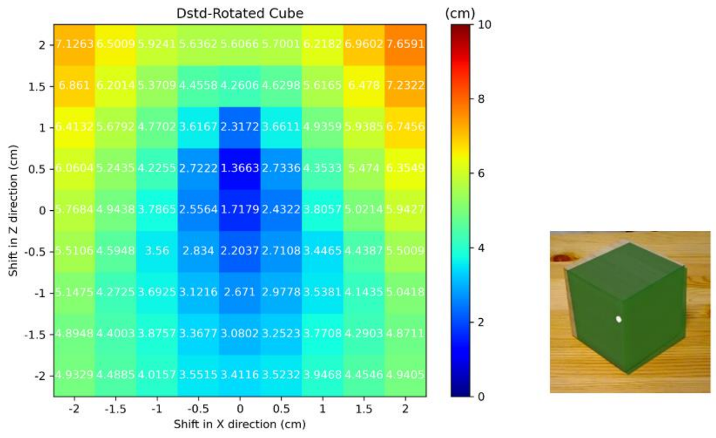
(a)



(b)



(c)



(d)

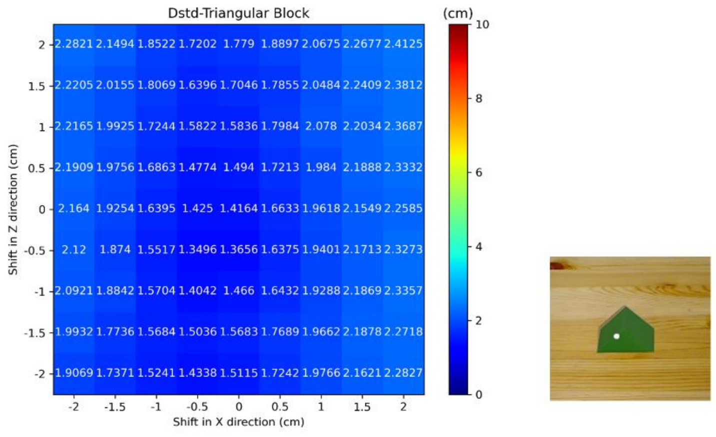


Fig. 2-34 Evaluation results using the D_{std} indicator

For the triangular block, the similar D_{ave} and D_{std} results in different shift positions are caused by its smaller size. In Fig. 2-35, when the virtual bear statue and the virtual triangular block both shifts left 2 cm to (-2, 0), their left parts do not overlap with their corresponding real objects. Thus, when we compare the depth maps of the scanned point cloud and the CAD model, d_i^{SCAN} related to that left part without overlapping will be the scanned points of the surrounding, i.e., the table surface. Because the bear statue is much higher than the triangular block, in the region without overlapping, the depth difference $\Delta d_i = \left|d_i^{SCAN} - d_i^{CAD}\right|$, the distance between the virtual bear statue and the table surface along the projection line, is larger than the depth difference of the triangular block case. The depth difference maps, maps of Δd_i, in Fig. 2-35 show that the largest Δd_i of the bear statue can be larger than 10 cm while most Δd_i of the triangular block are averagely 4 cm. Thus, the small Δd_i between the CAD model and the table surface cause that D_{ave} and D_{std} do not change apparently in different shift positions.

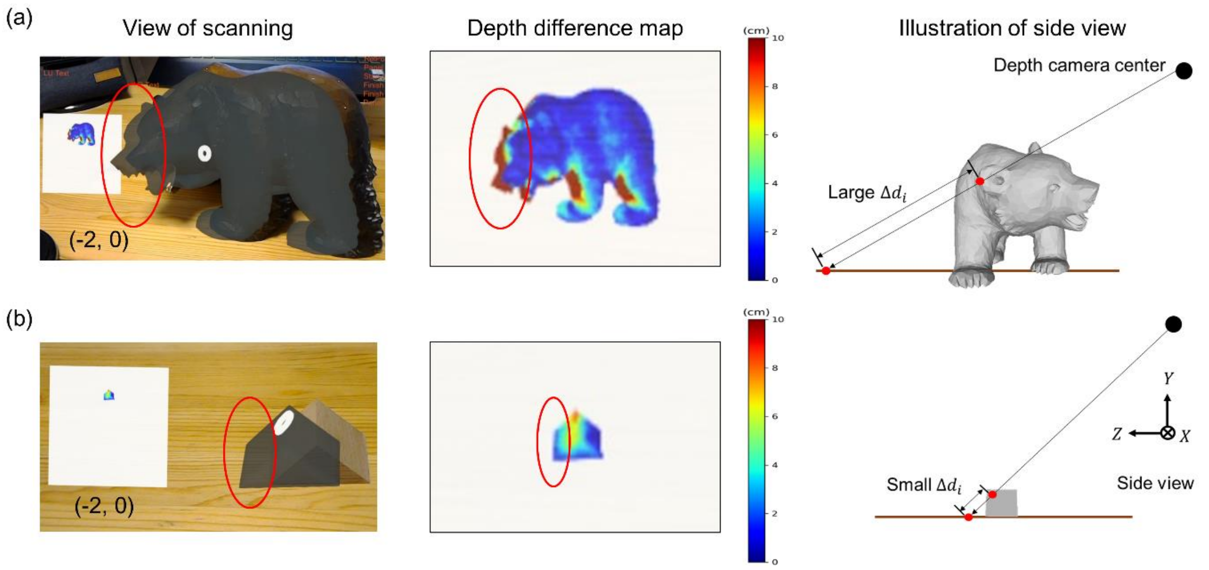
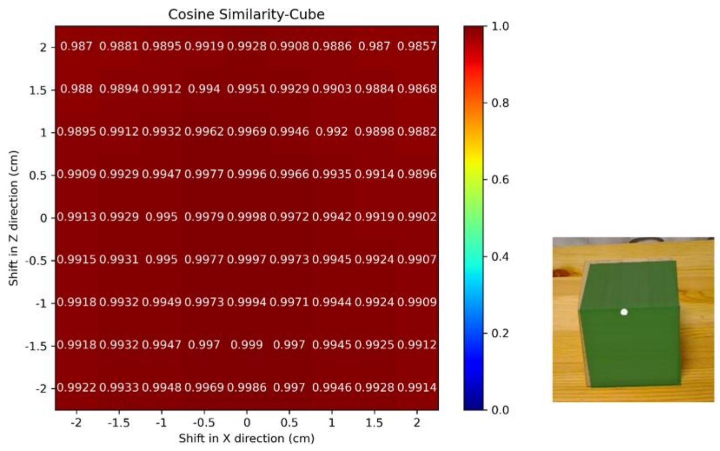


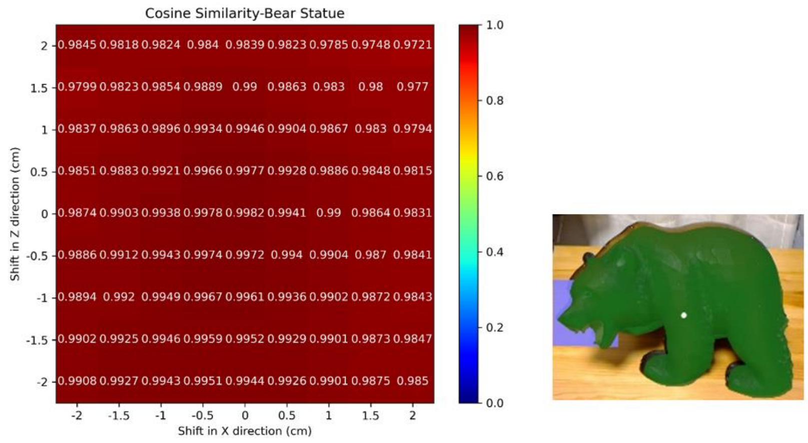
Fig. 2-35 Influence of the size of real objects to D_{ave} and D_{std}: compare (a) the bear statue and (b) the triangular block cases

Next, Fig. 2-36 and Fig. 2-37 show the results of D_{cs} and D_{ccs}, respectively. It can be seen that when the positions of the real and virtual objects are closer, both the D_{cs} and D_{ccs} indicators give larger values. It implies that the $\{d_i^{SCAN}\}_{i=1}^{n}$ and $\{d_i^{CAD}\}_{i=1}^{n}$ sequences are more similar.

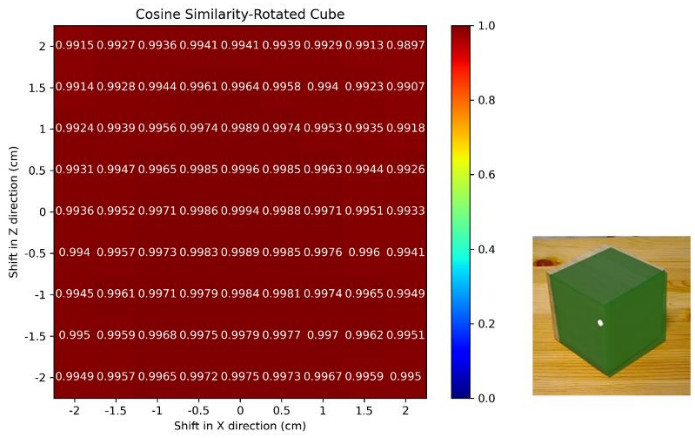
(a)



(b)



(c)



(d)

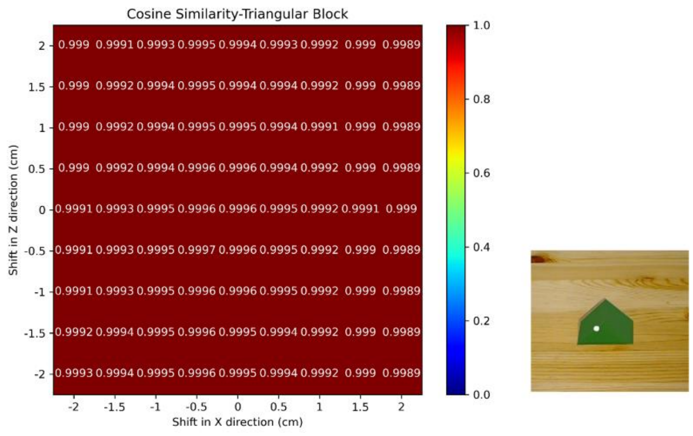
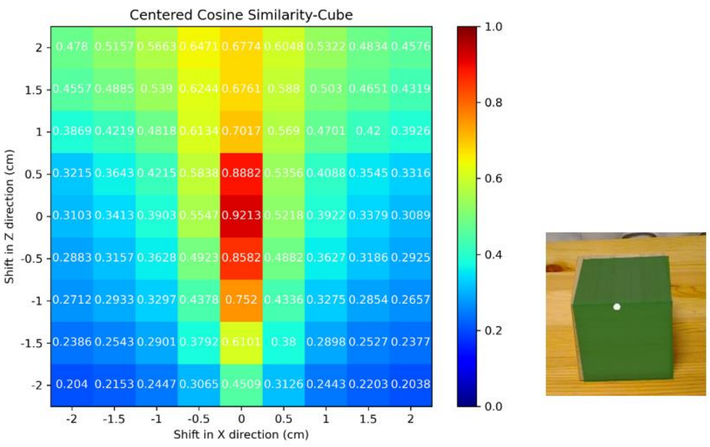
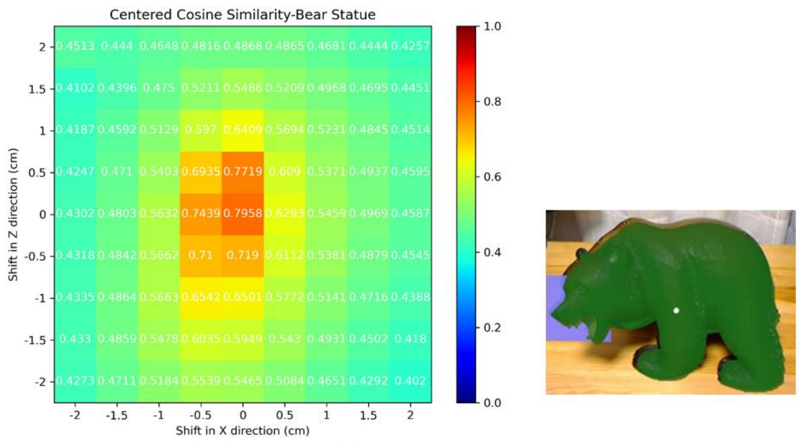


Fig. 2-36 Evaluation results using the D_{cs} indicator

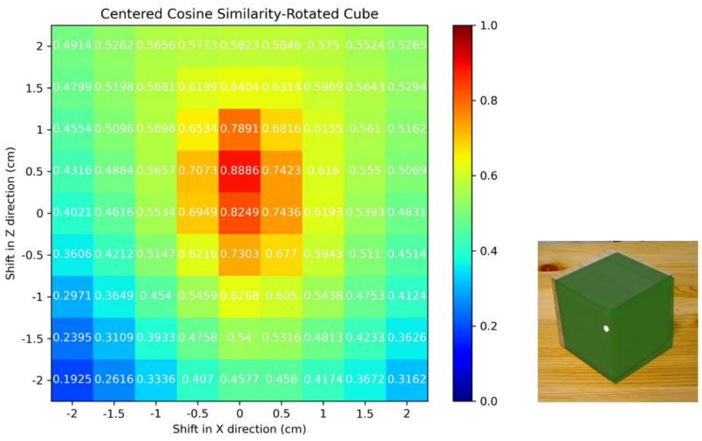
(a)



(b)



(c)



(d)

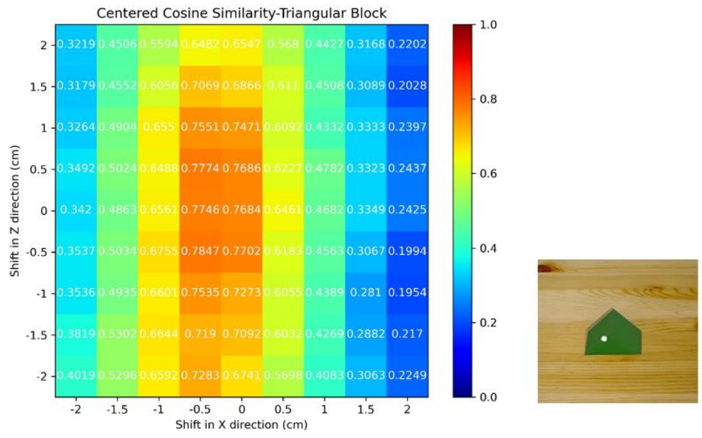


Fig. 2-37 Evaluation results using the D_{ccs} indicator

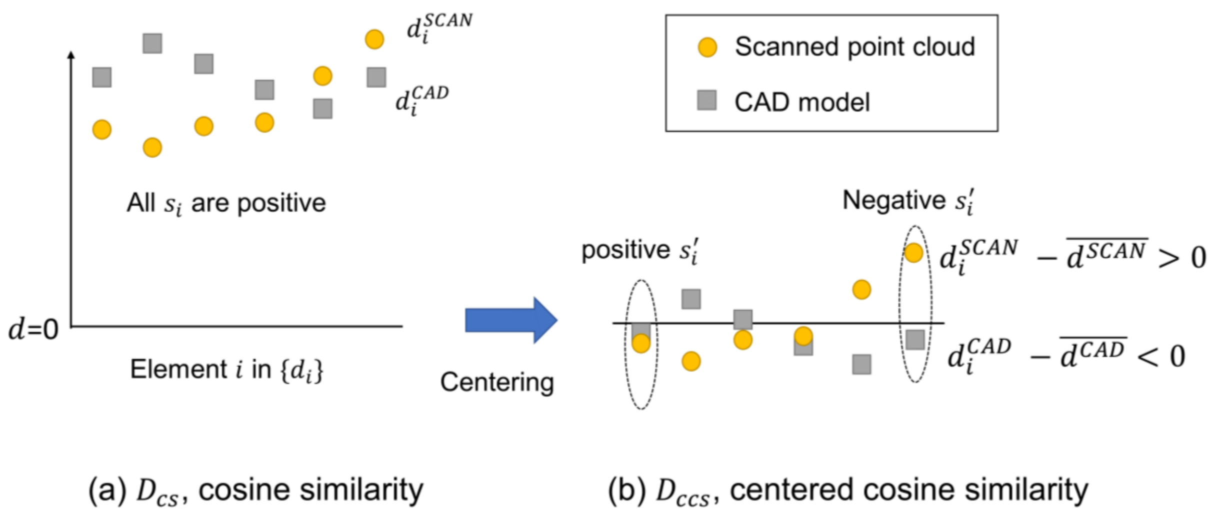
In Fig. 2-36, for all the four objects, the D_{cs} values in different positions vary slightly. For instance, by shifting the virtual bear statue from (−2, 0) to (0, 0), the D_{cs} value increases by only 1% as shown in Fig. 2-36(b). On the other hand, in Fig. 2-37(b), the D_{ccs} value increases by 85%.

Using the D_{cs} is not a proper indicator for us to compare the depth maps. According to Eq. (7), we can view the cosine similarity between d_i^{SCAN} and d_i^{CAD} as a sum of products $\{s_i\}_{i=1}^{n}$ such that

$$s_i = \frac{(d_i^{SCAN})(d_i^{CAD})}{\|\mathbf{d}^{SCAN}\|\|\mathbf{d}^{CAD}\|}.$$

Similarly, the centered cosine similarity is a sum of products $\{s_i'\}_{i=1}^{n}$ such that

$$s_i' = \frac{(d_i^{SCAN} - \overline{d^{SCAN}})(d_i^{CAD} - \overline{d^{CAD}})}{\|\mathbf{d}^{SCAN}\|\|\mathbf{d}^{CAD}\|}.$$



d_i^{SCAN}
d_i^{CAD}
● Scanned point cloud
■ CAD model
All s_i are positive
d=0
Element i in $\{d_i\}$
Centering
Negative s_i'
positive s_i'
$d_i^{SCAN} - \overline{d^{SCAN}} > 0$
$d_i^{CAD} - \overline{d^{CAD}} < 0$
(a) D_{cs}, cosine similarity
(b) D_{ccs}, centered cosine similarity

Fig. 2-38 Schematic plot of the computation of D_{cs} and D_{ccs}

In Fig. 2-38(a), because the output of the depth value is always positive, all s_i are positive, and the D_{cs} result keeps high even in different shift positions. On the other hand, for the computation of D_{ccs} in Fig. 2-38(b), after centering the sequences by their corresponding average depths, some s_i' are negative if d_i^{SCAN} and d_i^{CAD} are not both greater or less than their corresponding averages, $\overline{d^{SCAN}}$ and $\overline{d^{CAD}}$. This results in the lower D_{ccs} result than the D_{cs} when there is larger misalignment.

Based on the experiments, it can be seen that D_{ccs} is more sensitive to the misalignment between the real and virtual objects and is a better method for misalignment evaluation. In particular, for the case of the small triangular block, D_{ccs} changes more obviously in comparison with the other three indicators. In Sec. 2.3.3, we mention the shift or scaling of the scanned point cloud, as shown in Fig. 2-39. Because D_{ccs} is invariant to the shift and scaling, the constant shift and scaling errors of the scanned point cloud will not influence the D_{ccs} result while the errors influence the D_{ave} and D_{std} performance. Moreover, the output range of D_{ccs} is fixed in $[-1, 1]$. By setting a threshold of 0.72, when the D_{ccs} result is larger than 0.72, the position error, which is the misalignment between the real and virtual object, is within ±1 cm. This can be a preliminary indicator to judge whether the misalignment occurs. Therefore, with the comparison between different evaluation methods, we will use the centered cosine similarity D_{ccs} as the indicator to evaluate the misalignment between the real and virtual objects.

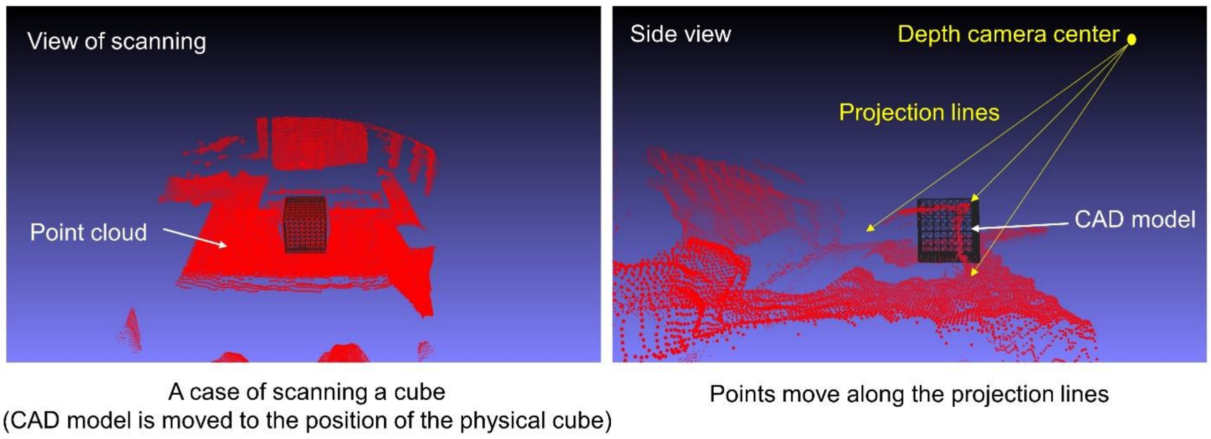

View of scanning
Point cloud
A case of scanning a cube
(CAD model is moved to the position of the physical cube)
Side view
Depth camera center
Projection lines
CAD model
Points move along the projection lines

Fig. 2-39 Shift or scaling problem of HoloLens-scanned point cloud

However, no matter which indicator we use, there is a problem that the indicator values are asymmetric. For example, in Fig. 2-37(a), indicator values at positions $(-1, 0)$ and $(0, 1)$ are 0.3903 and 0.7017, respectively. Indicator values changes at different rates as the object shifts along *X* and *Z* directions. This asymmetric problem derives from different variations of $\{d_i^{SCAN}\}_{i=1}^{n}$ along the two directions. It also means that the variations of depth differences $\{\Delta d_i\}_{i=1}^{n} =$

$\{\|d_i^{SCAN} - d_i^{CAD}\|\}_{i=1}^{n}$ are different. This derives from the influence of surroundings, the table surface, as the object moves in different directions.

In Fig. 2-40, let us see the front surface of the object. At position $(-1, 0)$, it can be seen that when comparing depth difference in the non-overlapping area, Δd_i is the distance difference between the table surface and the CAD model along the projection line starting from the depth camera center, i.e., the distance between points a and b $\|\overrightarrow{ba}\|$, and points c and d $\|\overrightarrow{dc}\|$. As comparing with the higher point, the depth difference becomes larger. The variation of Δd_i at position $(-1, 0)$ is schematically plotted in Fig. 2-41.

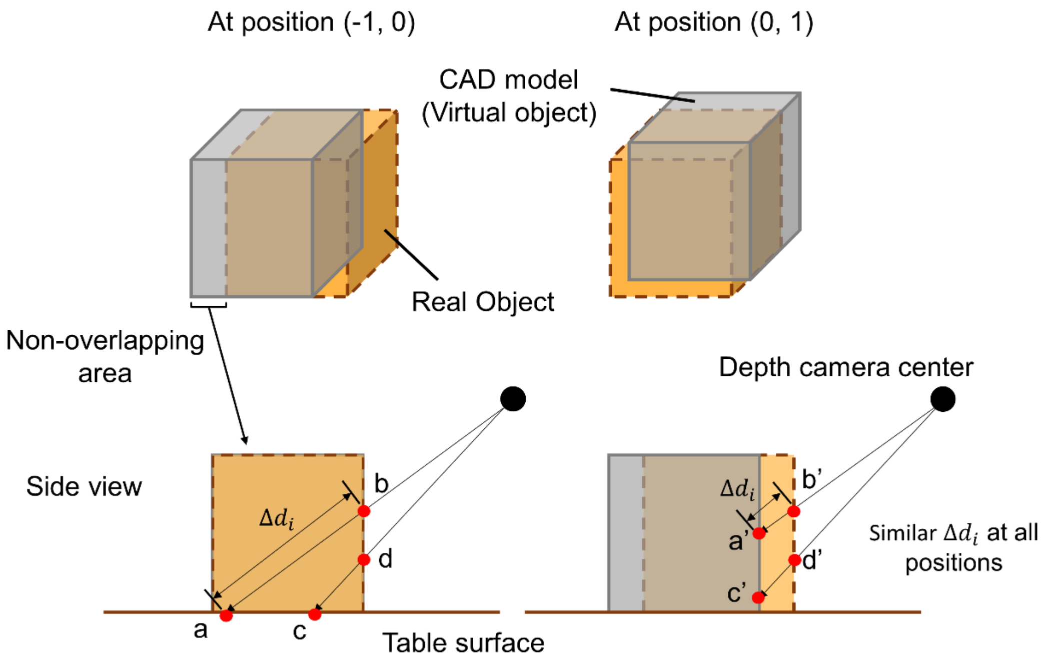


Fig. 2-40 Influence of surroundings on asymmetric indicator values

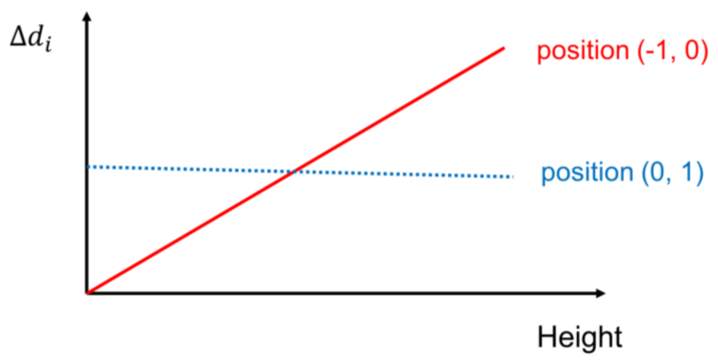


Fig. 2-41 Different variation of depth differences as the object is at different positions

On the other hand, at position (0, 1), by comparing the depth differences along the same projection lines, Δd_i is the distance between the point cloud of the real object and the CAD model; that is, $\left\|\overrightarrow{b'a'}\right\|$ and $\left\|\overrightarrow{d'c'}\right\|$. We see that $\left\|\overrightarrow{b'a'}\right\|$ is approximately equal to $\left\|\overrightarrow{d'c'}\right\|$ in comparison to the difference between $\left\|\overrightarrow{ba}\right\|$ and $\left\|\overrightarrow{dc}\right\|$. The depth difference does not change so much as comparing with the higher point as shown in Fig. 2-41. Since the variations of depth differences in the two cases are different, it leads to the difference of calculated indicator values. For the case of position (−1, 0), the existence of surrounding objects influences the variation of calculated depth difference and further influence the output indicator value. To make the indicator value symmetric, it is a possible way to perform object tracking to remove the pixels of surroundings from the depth map of the real object.

2.6 Demonstrations of HMD-based AR assembly assistance system

2.6.1 System Implementation

For the implementation of the AR assembly assistance system, we use the Unity software (version 2017.4) [44] to develop user interfaces and necessary functions. Unity is commonly used as the software for game development, and it can also support the development of AR applications for the HoloLens.

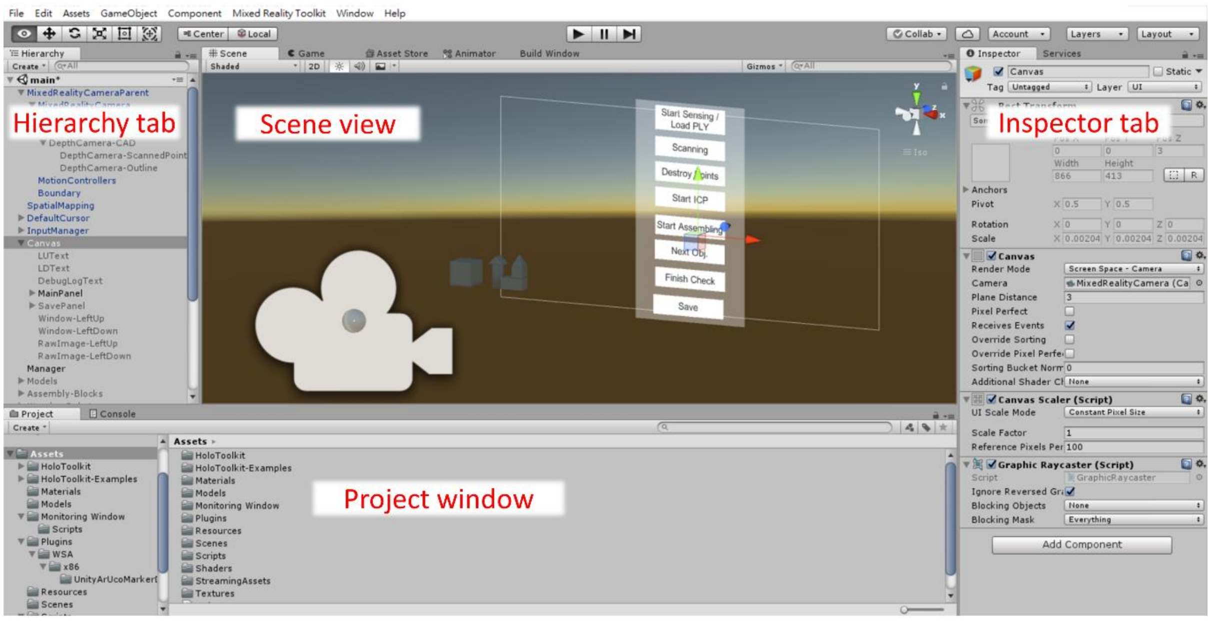


Fig. 2-42 Introduction to Unity interface for application development.

Fig. 2-42 shows the development interface in Unity. It consists of four regions: project window, hierarchy tab, scene view, and inspector tab. In the project window, we can manage files for application development, including CAD models of an assembly, script files telling how the system and the virtual objects behave, and Mixed Reality Toolkit developed by Microsoft. Hierarchy tab shows the list of all active objects in the scene view. The scene view visualizes how the system will look like. We can see the CAD models and design a graphical user interface for the system. Finally, in the inspector tab, we can control the position and orientation of the CAD models and apply developed script files to the CAD models to control how the CAD models behave. For example, we apply a script called “TwoHandManipulatable.cs” (from Mixed Reality Toolkit) to the reference virtual object and make the virtual object can be manipulated by user's hands. This script takes the movement of the user's hands as input to control the position and the orientation of the virtual object. This realizes the rough alignment of coordinate calibration in Sec. 2.4. In addition, script files in Unity are developed using C# programming language. Next, Fig. 2-43 introduces the relationship between the designed view and the practical view of the system.

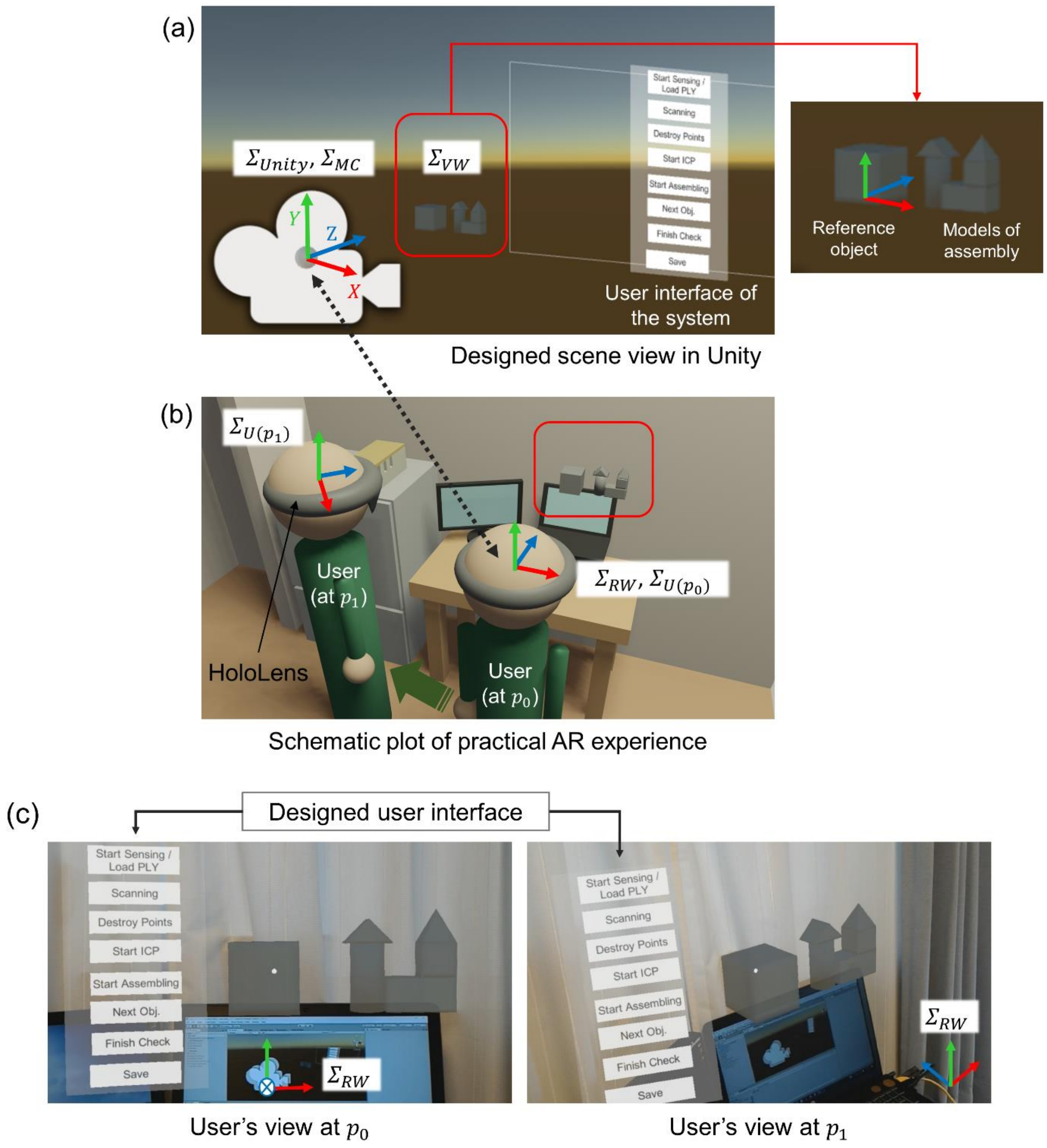


Fig. 2-43 Relationship between the designed system and the practical user experience in AR: (a) designed view in Unity, (b) a schematic plot of user experience in AR, and (c) user's view of a real scene at different positions.

For the system development in Unity, in Fig. 2-43(a), it mainly contains three types of objects: cameras, CAD models, and a user interface for the control of the assembling process. Those objects are defined in the coordinate system of Unity, Σ_{Unity}, which corresponds to the coordinate system of the real world, Σ_{RW}, as shown in Fig. 2-43(b).

Set the main camera

For camera objects, especially, a camera object called *Main Camera* in Unity handles head tracking and stereoscopic rendering of the HoloLens. That is to say, the coordinate system of the main camera Σ_{MC} corresponds to the coordinate system of the user Σ_{U}. Moreover, because Σ_{MC} (Σ_{U}) moves with the movement of the user's head, the starting position of the user can be set by setting the position of the main camera. By setting the main camera position to (X: 0, Y: 0, Z: 0) in Σ_{Unity}, the starting position of the user (user's position at p_0 in Fig. 2-43(b)) will be the same as the origin of Σ_{RW}. It can be said that when the developed AR system initially starts up, the user's initial position in the real world determines where the origin of Σ_{RW} is.

Define the virtual world

Next, we define a local coordinate system called as the virtual world Σ_{VW}, which includes the reference virtual object and the CAD models of the assembly. The origin of Σ_{VW} is defined at the reference virtual object. In addition, the CAD models of the assembly are defined in Σ_{VW}. Thus, when the reference virtual object moves, the CAD models of the assembly also moves the same way. Moreover, by setting the position of the reference virtual object at (0, 0, 1) in Σ_{Unity}, which corresponds to (0, 0, 1) in Σ_{RW}, it will render the reference object 1 meter in front of the user's starting position.

Fig. 2-43(c) shows the user's view through the HoloLens at p_0 and p_1, as marked in Fig. 2-43(b). With the above position setting of the main camera and of the reference object, when the user wears the HoloLens and starts up the developed AR system at p_0, the user can see the virtual objects 1 meter forward. Then, the user moves to p_1 and can see that the virtual objects are fixed at the same position.

Design a user interface

Furthermore, we design a basic user interface to help the user control the system. It contains a few buttons to let the user trigger different functions which are realized in script files. Table 2-2 lists and introduces the essential buttons to control the assembling process. The position of the buttons in the user interface is listed according to the execution sequence from top to bottom. More detailed relationship between the designed function buttons in the user interface and the system process is shown in Fig. 2-44

Table 2-2 The function of the buttons in the designed user interface

Button name	**Function**
Start Sensing	Enable the depth camera. The depth camera will start scanning the physical environment.
Scanning	Record and visualize the point cloud scanned by the depth camera.
Start ICP	Take the reference virtual object and the scanned point cloud as input and perform the point-to-plane ICP algorithm.
Start Assembling	Show the CAD model of the first part in the assembly and evaluate misalignment
Next Object (Obj.)	Show the CAD model of the next part and evaluate misalignment
Finish Check	Finish the assembling process and the misalignment evaluation.

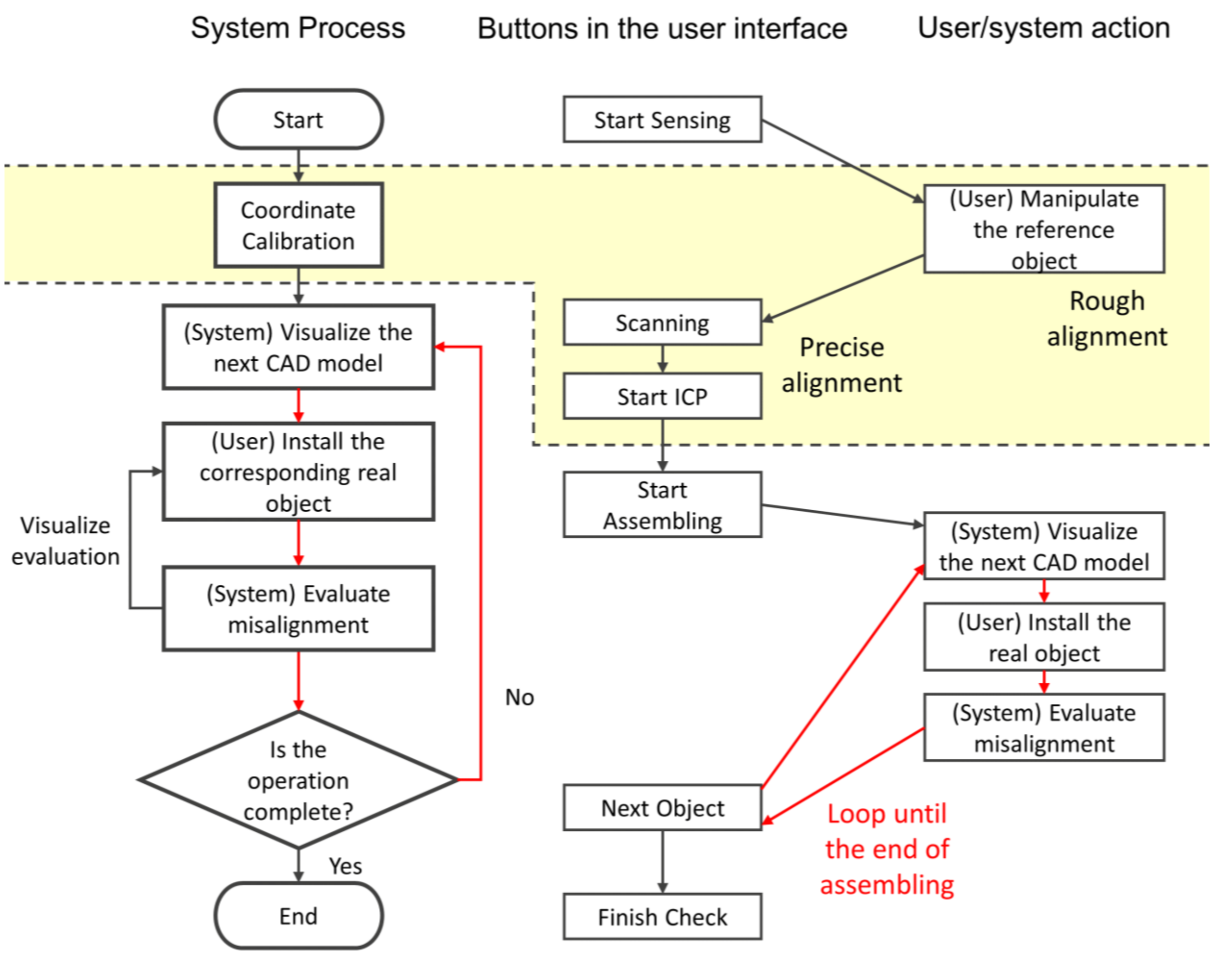

System Process
Buttons in the user interface
User/system action
Start
Start Sensing
Coordinate Calibration
(User) Manipulate the reference object
Rough alignment
(System) Visualize the next CAD model
Scanning
Start ICP
Precise alignment
(User) Install the corresponding real object
Visualize evaluation
(System) Evaluate misalignment
Start Assembling
(System) Visualize the next CAD model
(User) Install the real object
(System) Evaluate misalignment
No
Is the operation complete?
Yes
End
Next Object
Finish Check
Loop until the end of assembling

Fig. 2-44 Relationship between designed interface and system process

Set virtual depth cameras

In Sec. 2.5, to compare the depth maps of the virtual object and of the scanned point cloud for misalignment evaluation, two virtual cameras are set to render the two depth maps, respectively. For the implementation, in Unity, we first set two camera objects at the same position as the physical depth camera.

Since the physical depth camera is fixed in the HoloLens and moves with the user's head, the position of the depth camera relative to the user is fixed. The relative orientation is also fixed. Thus, we can set the virtual cameras in Σ_{MC}, which corresponds to Σ_U with fixed position and orientation values. Fig. 2-45 shows the coordinate system of the virtual camera Σ_{VC} related to Σ_{MC}. With our preliminary evaluation, the origin of Σ_{VC}, the center of the virtual camera, is at (0 m, 0.0438 m, 0.0663 m) in Σ_{MC}. Moreover, Σ_{VC} is oriented by 33.5° along the X axis clockwise according to the left-hand rule. To determine the values of the position and the orientation, we refer to the transformation matrix $\mathbf{M}_{U\leftarrow C}$ between the coordinate systems of the user Σ_U and of the physical depth camera Σ_C, which is recorded in the HoloLens as introduced in Sec. 2.3.3. Through the transformation matrix between Σ_U and Σ_C, we can evaluate the origin of Σ_C in Σ_U, which corresponds to the origin of Σ_{VC} in Σ_{MC}. The orientation can be evaluated by determining the direction of the axes of Σ_C (Σ_{VC}) in Σ_U (Σ_{MC}).

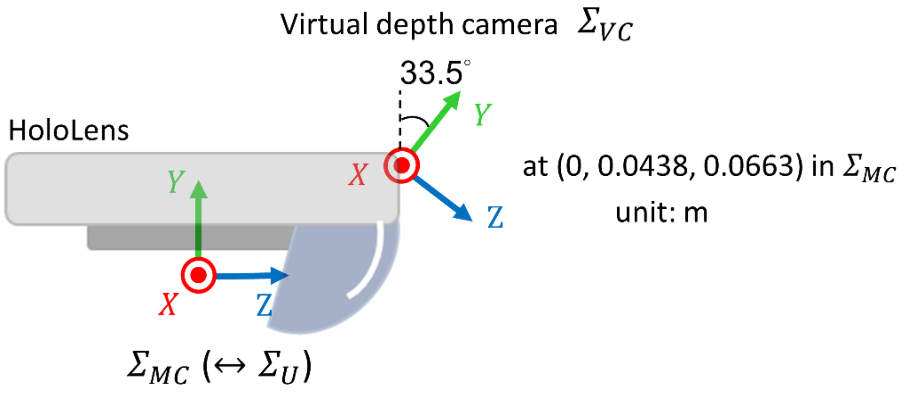

Virtual depth camera Σ_{VC}
33.5°
Y
HoloLens
X
Y
Z
at (0, 0.0438, 0.0663) in Σ_{MC}
unit: m
X
Z
Σ_{MC} ($\leftrightarrow$ Σ_U)

Fig. 2-45 Schematic plot of the relationship between Σ_{MC} and Σ_{VC} in Unity

According to Eq. (3), a point $\mathbf{v}$ described in Σ_U and Σ_C is represented by 3×1 vector $\mathbf{v}_U$ and $\mathbf{v}_C$, respectively. The 4×4 homogeneous transformation $\mathbf{M}_{U\leftarrow C}$ can be divided into a 3×3 rotation matrix $\mathbf{R}_{U\leftarrow C}$ and a 3×1 translation matrix $\mathbf{T}_{U\leftarrow C}$. Thus, the equation of the transformation between Σ_U and Σ_C can be written as:

$$\begin{bmatrix}\mathbf{v}_U\\1\end{bmatrix} = \mathbf{M}_{U\leftarrow C}\begin{bmatrix}\mathbf{v}_C\\1\end{bmatrix}$$

or

$$\mathbf{v}_U = \mathbf{R}_{U\leftarrow C}\mathbf{v}_C + \mathbf{T}_{U\leftarrow C} \tag{9}$$

By substituting $\mathbf{v}_C = (0,0,0)^T$ into Eq. (9), we obtain $\mathbf{v}_U = \mathbf{T}_{U\leftarrow C}$, so the position of the virtual depth camera in Σ_U (Σ_{MC}) is set as $\mathbf{T}_{U\leftarrow C}$. With the same way, we can substitute $\mathbf{v}_C = (1,0,0)^T$, $\mathbf{v}_C = (0,1,0)^T$, and $\mathbf{v}_C = (0,0,1)^T$ into Eq. (9) to obtain vectors of the coordinate axes of Σ_C in Σ_U. They can then be used to determine the orientation angle of the depth camera.

After determining the position and the orientation of the virtual camera, the next is to tell the GPU in the HoloLens to generate the depth map from the view of the virtual camera. In default, the camera object in Unity renders the view as an RGBA image. To generate the depth map, we need to write and apply a script file to make the virtual camera render the depth map by setting the camera mode to "DepthTextureMode.Depth" [45].

Visualize the evaluation result of misalignment

With such designs, we next preliminarily investigate the performance of the developed system through the following three demonstrations. These demonstrations aim to verify basic functionality of the system. Considering the quality of point clouds generated by the depth camera in HoloLens, we choose physical 3D objects whose one of the dimensions is at least larger than 3 cm so that point cloud data can keep more details of the objects. Meanwhile, we also choose objects that are not dark, not transparent, and not shiny to avoid poor scanned point clouds. Moreover, to clearly visualize the evaluation result of misalignment between the real and virtual object, we color the virtual object according to the calculated D_{ccs}, and the color change is shown by a color bar in Fig. 2-46. When the real and virtual objects are matched, the D_{ccs} will be close to one, and the virtual object will be colored by blue; otherwise, it will be colored by red. Moreover, we also design a mechanism of exception that the virtual object will be colored by white if the percentage of depth differences $\{\Delta d_i\}_{i=1}^{n}$ that $\Delta d_i > 10$ cm is more than 15%. When there is an occlusion problem, this can be used to notify the user to change to other views for misalignment. Or when the user places an object whose volume is much different to the target object, this can be a way to warn the user, and the system will be no need to compute the D_{ccs} value.

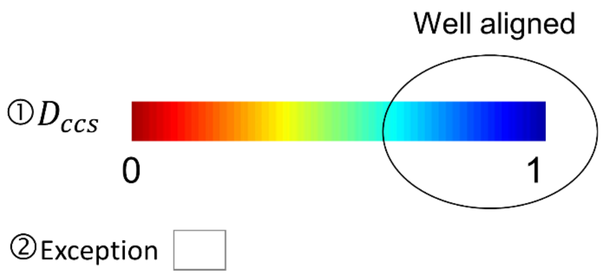


Fig. 2-46 Color bar for showing misalignment evaluation results: (1) color the CAD model according to the calculated D_{ccs} value if the percentage of depth differences

that $\Delta d_i > 10$ cm is less than 15% and (2) color the CAD model with white if the percentage of depth differences that $\Delta d_i > 10$ cm is more than 15%

2.6.2 Experiment 1: Assemble primitive wooden blocks

To preliminarily evaluate system performance, we first conduct an experiment that aligns wooden blocks on a table to verify basic functions. In Fig. 2-47(a), a wooden cube with an edge of 10 cm is used as a reference object, and the corresponding reference virtual object can be found at the origin of the virtual world in the CAD model, as shown in Fig. 2-47(b). Moreover, five primitive wooden blocks of different shapes-a cylinder, a square prism, a triangular block, a smaller cube, and a pyramid-are used for assembly operation and should be installed in the order that is designed and numbered in Fig. 2-47(b). The process of the assembling operation is shown in Fig. 2-48. At step 0, coordinate calibration is performed to define the working area first. Then, from step 1 to 5, the system displays the virtual object in the designated order, and the user puts the corresponding real object to that position. Finally, step 6 shows the completeness of the operation.

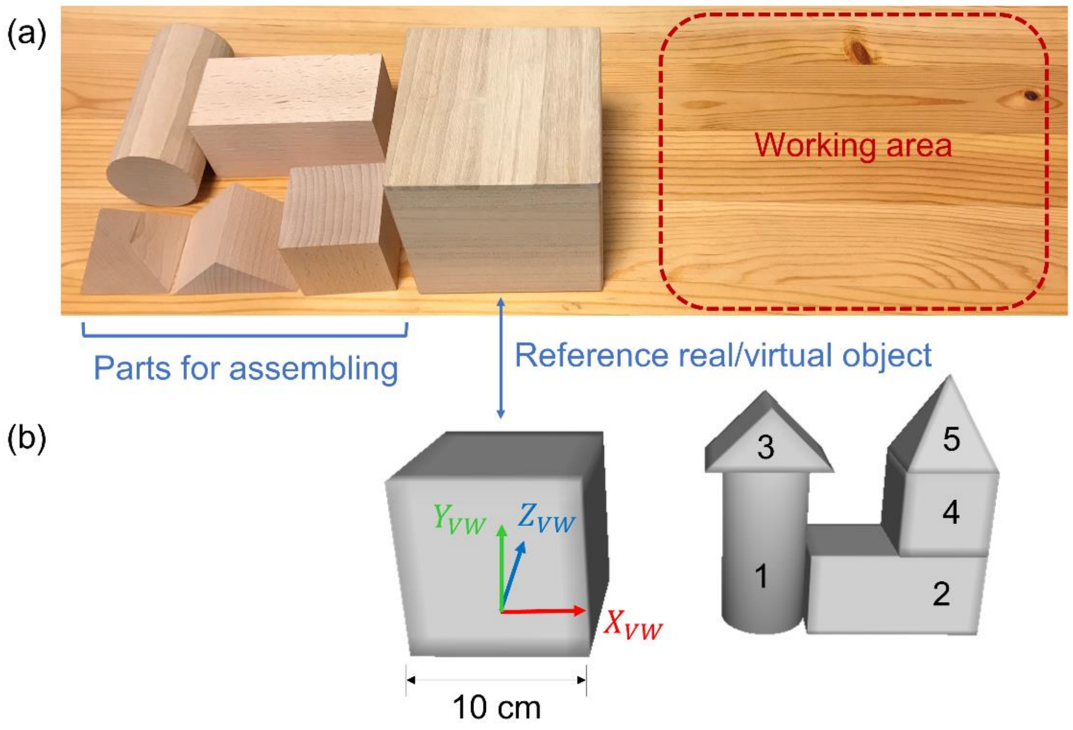


Fig. 2-47 Experimental setup of a block assembly: (a) real objects and (b) designed alignment of parts in the CAD model

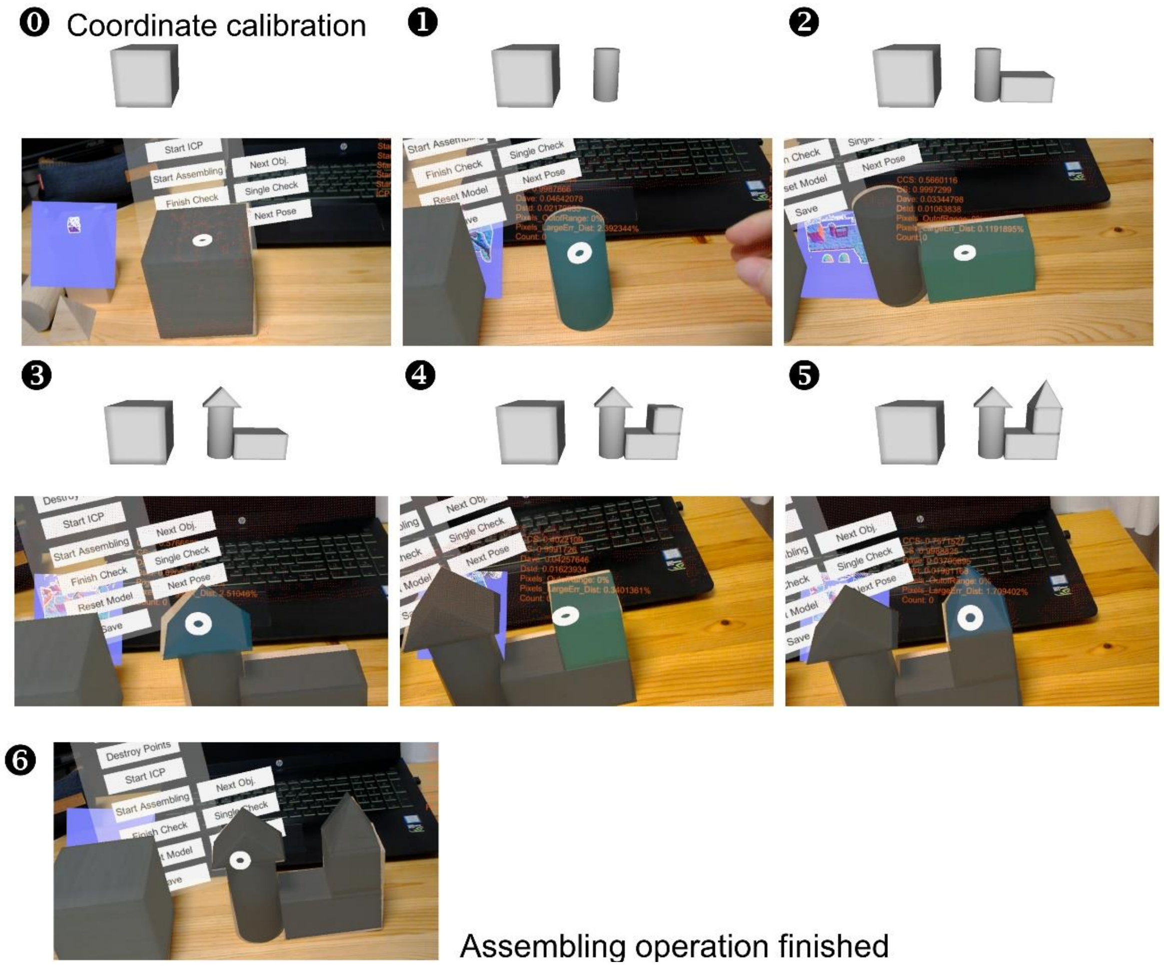


Fig. 2-48 The process of assembling blocks: put blocks to the designated positions

During the assembling process, from step 1 to 5, the system evaluates misalignment between the real and virtual objects by the centered cosine similarity method at approximately 30 fps. The user can move the real object to the position of the virtual object and simultaneously check whether the color of the virtual object is close to blue, which indicates well aligned, or not. However, even the real and virtual objects are matched with the error smaller than 1 cm, the evaluated D_{ccs} value (or the color of the virtual object) sometimes varies dramatically between 0.5 and 0.9. The reason may derive from the update of the point cloud scanned from different views and the change of depth values of the pixels that belongs to the outline of the object in the depth map of the scanned point cloud.

2.6.3 *Experiment 2: Assemble a wooden robot*

Next, we conduct an experiment of assembling a wooden robot with a certain pose. In Fig. 2-49(b), the robot is designed to do a handstand and stretch its legs back and forth. The robot contains five parts: a body, two hands, and two legs. There are also four screws to fix the hands and legs to the body, and each screw is put into the part in advance.

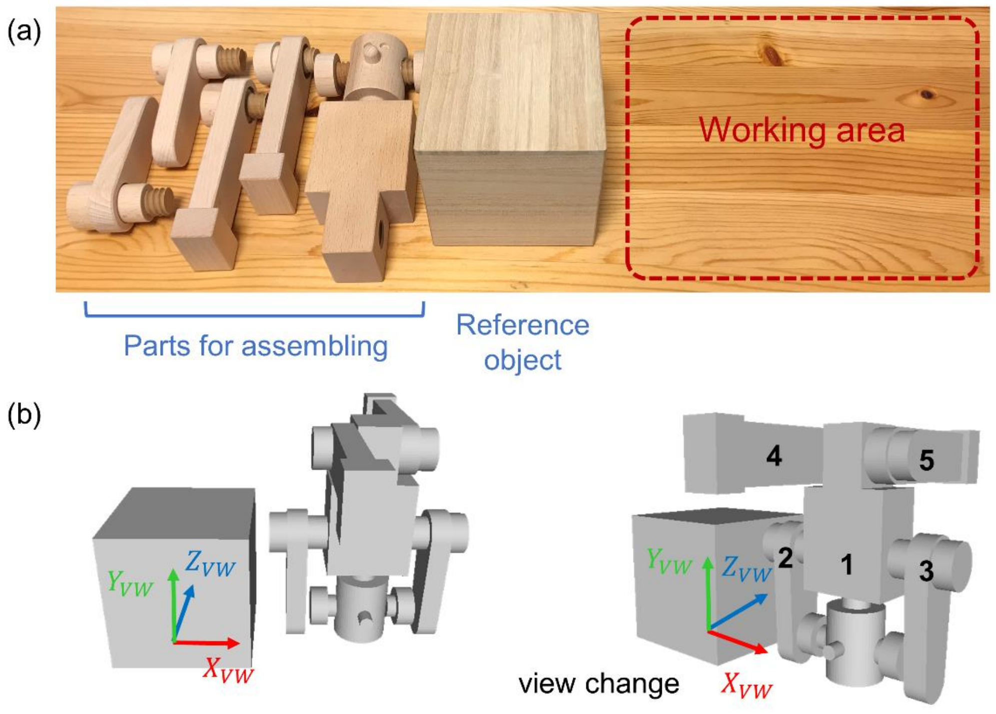


Fig. 2-49 Experimental setup of a wooden robot assembly: (a) real objects and (b) designed alignment and pose in the CAD model

Fig. 2-50 shows the assembling process. Identically, step 0 is coordinate calibration using the wooden cube with an edge of 10 cm. Step 1 to 5 shows the assembling of each part. From step 2 to 5, parts of the hands and legs are screwed to the body, and we put the assembly to the position of the virtual object to check if there is an assembly error. Step 6 shows the completeness of the assembly.

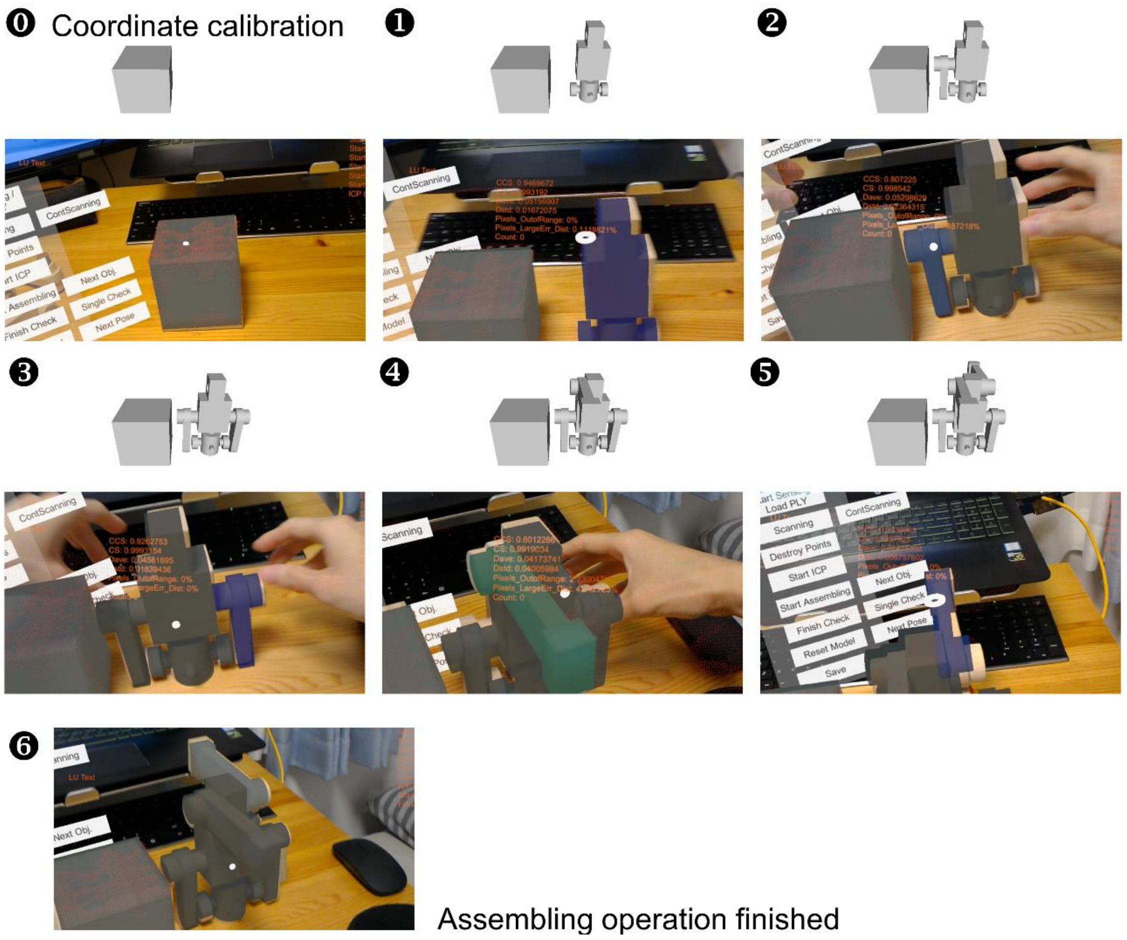


Fig. 2-50 The process of assembling a wooden robot with the designated pose

When assembling the leg parts, in steps 4 and 5, the user follows the evaluation results and the orientation of the virtual leg to adjust the physical leg part to the designed orientation, as shown in Fig. 2-51. Fig. 2-52(a) shows that there is a gap of 2mm between the leg part and the body, and in the design, the included angle of the leg part and the body line should be a right angle. However, due to the existing of the gap, if the user does not adjust the leg part and tighten the screw, the top of the leg part will move downward by 6 mm, as shown in Fig. 2-52(b). The assembly result will not be as designed. This shows that the misalignment evaluation can play a role in warning the assembly error, and in this case, the developed evaluated method can detect the displacement of 6 mm.

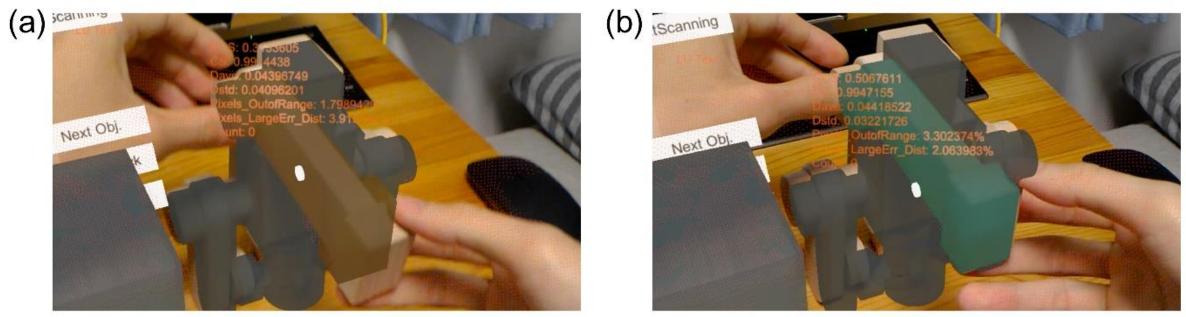


Fig. 2-51 Detection of an assembly error that may be overlooked easily: (a) misalignment is detected, and (b) no misalignment is detected after adjustment

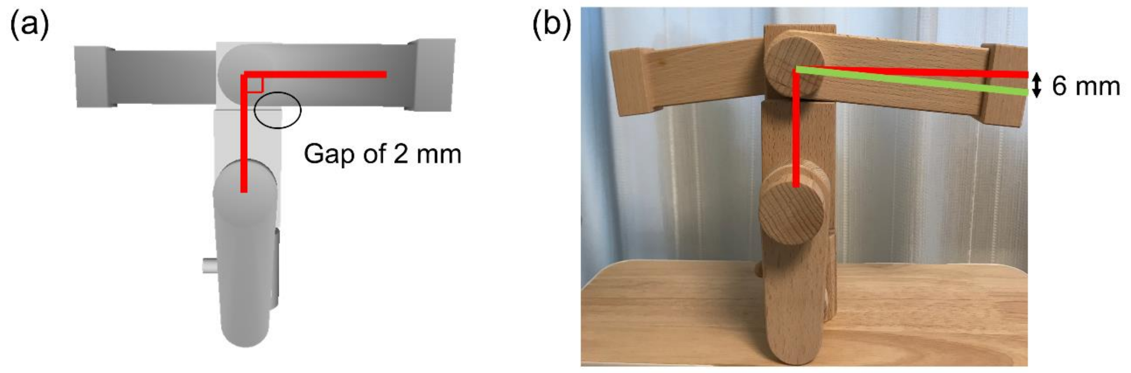


Fig. 2-52 Differences between (a) designed pose in the CAD model and (b) physical assembly without careful assembling

2.6.4 Experiment 3: Arrange decorations in a room-scale environment

Finally, we make the user arrange decorations in a room-scale environment. This is a preliminary experiment to evaluate whether the system can assist the user in doing assembling in room-scale environments. The user who wears HoloLens can keep moving and is required to place decorations at designated positions. In Fig. 2-53, we prepare three statues of a wooden robot, a frog, and a bear and would like to place them at different locations in an 8 m^2 room. The cube with an edge of 10 cm is used as the reference object to determine the origin of the virtual world. The reference object and the three statues are numbered to show the sequence of installation.

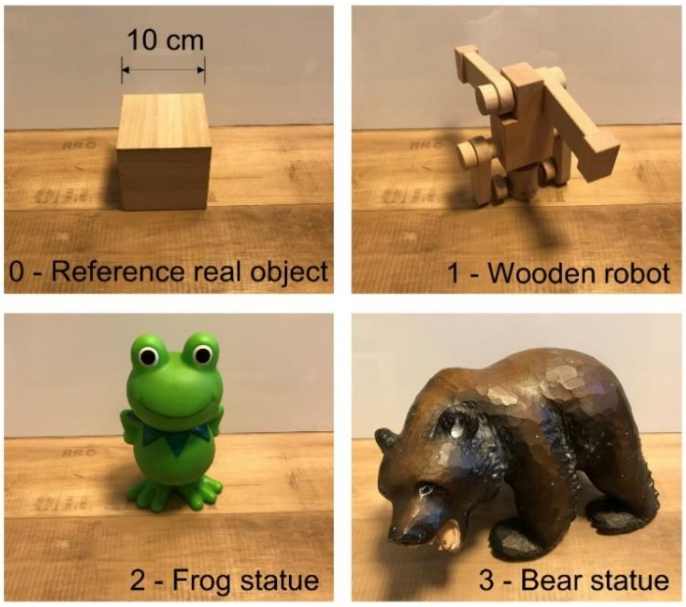


Fig. 2-53 A reference real object and decorations for arrangement. The sequence of installation: (1) wooden robot, (2) frog statue, and (3) bear statue.

Fig. 2-54 shows the target room with the decorations in the designated positions. A schematic plot of the room and the decorations are also plotted to explain the detailed process. Notice that the wooden robot is hidden by another object.

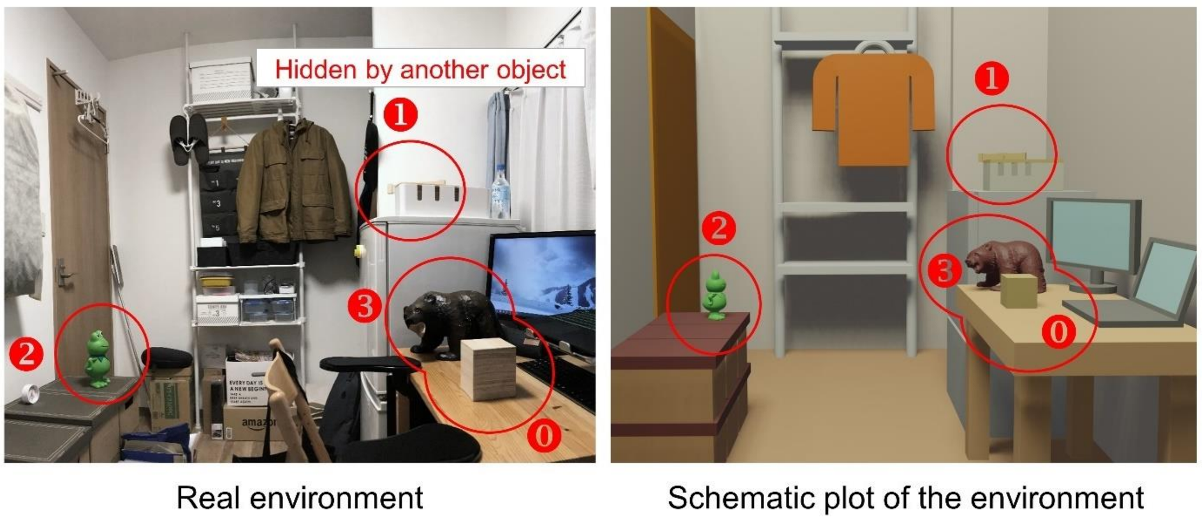


Fig. 2-54 Target room environment for decoration arrangement

To construct arrangement information for the user, first, we have to design and record the arrangement of the statues in the room in advance. A designer wears the HoloLens and moves the CAD models of the decorations to desired positions by hand manipulation. Fig. 2-55(a) shows the process of the arrangement design.

Through the HoloLens, the designer can see the decorations and the reference virtual object. First, at step 0, the reference virtual object is moved to the position of the corresponding reference real object by rough alignment and precise alignment in sequence to define the origin of the virtual world in the real world. Next, the CAD models of the three decorations are moved to different positions in the room. Their positions in the virtual world are shown in Fig. 2-55 and recorded as information for the user to place the decorations.

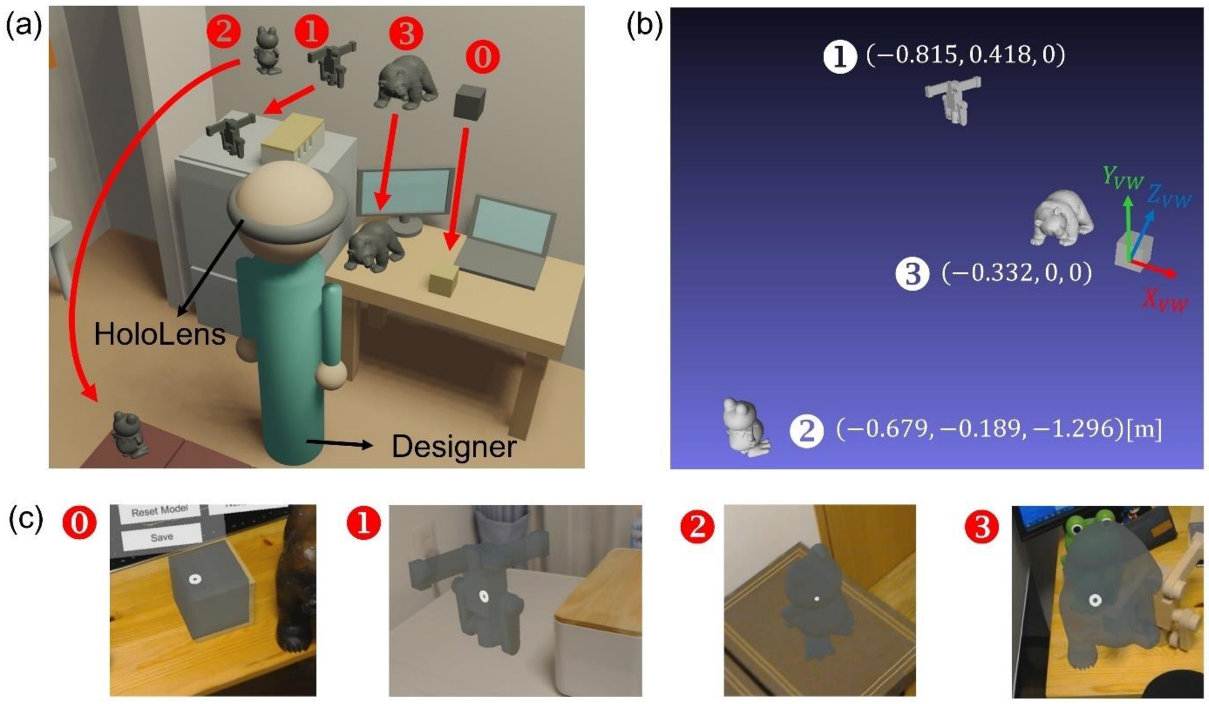


Fig. 2-55 Design decoration arrangement: (a) A schematic plot of placing the CAD models of the decorations for arrangement design, (b) The position of the CAD models in the virtual world, and (c) Results of designed arrangement seen through the HoloLens

Decoration arrangement

After the construction of arrangement information, the user can wear the HoloLens and follow indications to place the decorations step by step. Fig. 2-56 shows the process of decoration arrangement. Initially, the user performs coordinate calibration by aligning the reference object. Then, the system shows the CAD model of the decoration one by one at the designed locations. The scene that the user sees through the HoloLens will be the same as shown in Fig. 2-55(c), which the results of designed arrangement by the designer. The user places the corresponding physical statues to their position in sequence, and the system evaluates misalignment during arrangement process. Fig. 2-57 shows the mixed reality photos of the arrangement results.

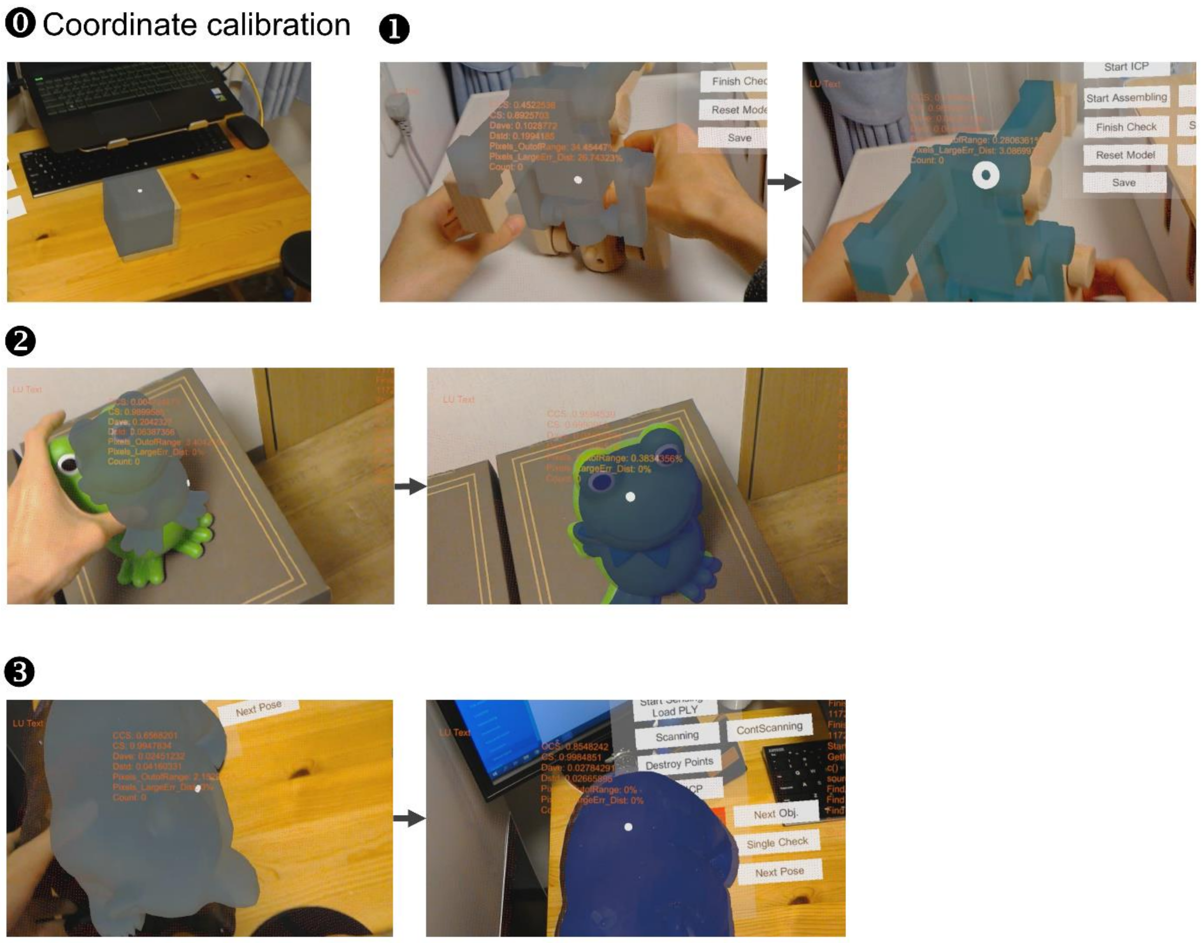


Fig. 2-56 The process of decoration arrangement with the designated pose

Through the experiment, the user follows the indication and place the statues to their locations within the accuracy of ±1 cm. Misalignment evaluation helps the user place the statues to their position faster. Without misalignment evaluation, to check whether the statues are placed well at designated positions or not, the user has to observe the statues from different views back and forth. Moreover, the three decorations and the reference object are inside a cubic space that has a dimension of 1.3 m × 0.8 m × 0.6 m. It indicates that the system is possible to support the user to assist an assembly that has such volume in an 8 m^2 room and can detect misalignment with the accuracy of ±1 cm. In addition, in this room environment, it is hard to take a mixed reality photo that shows all the virtual objects for arrangement because of the limitation of the space and insufficient field of view of the RGB camera in the HoloLens. Although the user can still place the decorations according to the information as shown in Fig. 2-55(b), the user has to measure the distance between the decoration and the reference object to verify the installation location. Nevertheless, the AR glasses system can directly show the installation locations in the real scene of the user's view. Thus, the user focuses on placing the decoration to the virtual object's position without considering the distance between the decoration and the reference object.

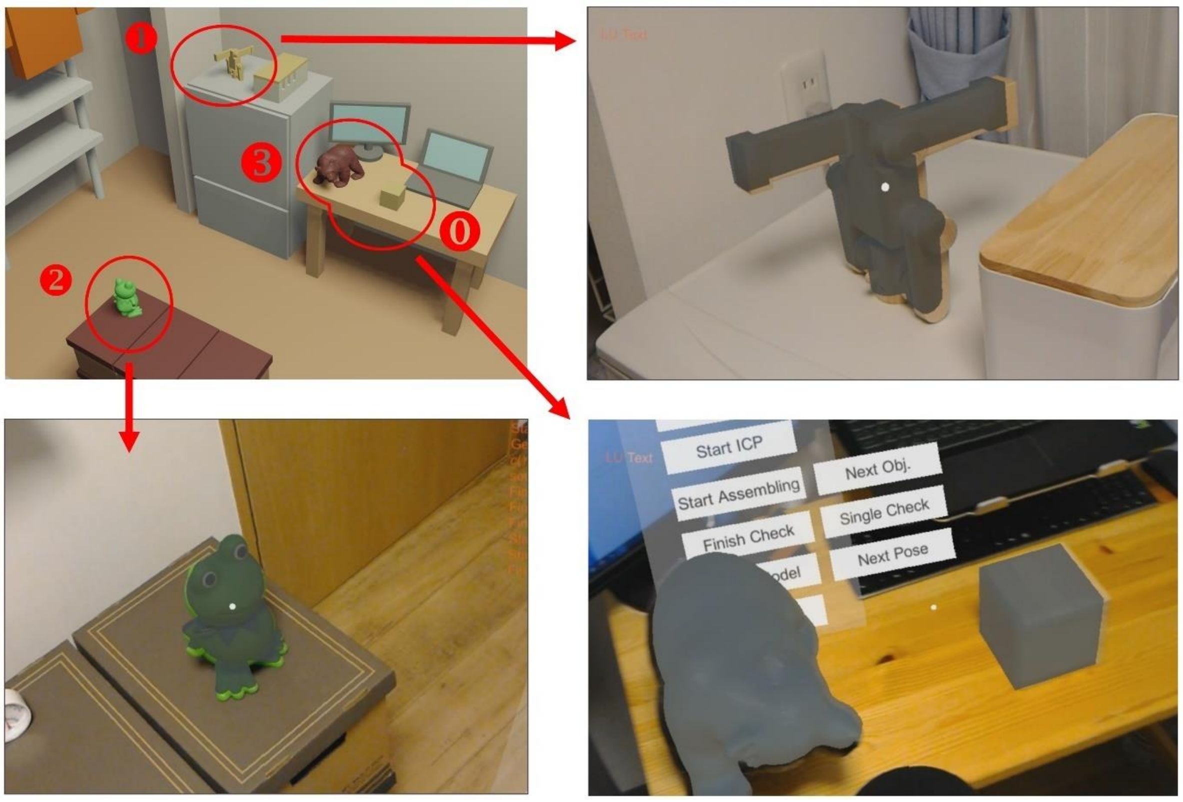


Fig. 2-57 Mixed reality photos of the results of decoration arrangement

2.7 Summary

Using AR technology is a potential way to ensure better experience during the process of assembling a product. Besides assembly instruction from an AR-based system, it is important to determine whether assembly errors occur. In this study, we propose two methods to construct an essential AR-based assembly assistance system using only an HMD, in this case, Microsoft HoloLens: (1) coordinate calibration and (2) efficient evaluation of misalignment between real and virtual objects.

For coordinate calibration, with the function of positioning the user in the real and world and point cloud data from the HoloLens, we can achieve coordinate calibration without using AR markers to show CAD models of parts at designated locations. This is achieved by aligning the reference virtual object to the reference real object. We design the alignment process with two steps: rough alignment (user′s hand manipulation) and precise alignment (point-to-plane ICP) performed in sequence. Rough alignment can be performed fast, but the alignment accuracy is in centimeter level. On the other hand, precise alignment can achieve accuracy in millimeter level, but the initial position of the reference virtual object needs to be close to the reference real object. The reference real object should be inside the bounding box of the reference virtual object. Otherwise, the precise alignment result will converge to a local minimum. Thus, by integrating rough and precise alignment, we can achieve coordinate calibration in millimeter level.

Moreover, for evaluation of any misalignment between virtual and real objects, with our preliminary implementation, the system compares depth maps of the real and virtual objects to evaluate misalignment at approximately 30 fps, which will not influence user′s experience apparently. Thus, the user can rapidly understand the installing condition. If an assembly error occurs, the system can instantly warn the user by coloring the CAD model. Based on experimental results of different evaluation methods, we use the centered cosine similarity method as an indicator to compare the depth maps and perform evaluation, which is a more robust method to the errors of point cloud data of the physical depth camera.

To sum up, with the proposed methods, we can make good use of functions and raw sensor data provided by the HoloLens to construct an essential HMD-based AR assembly assistance system without using any AR marker and external devices to define the transformation relationship between the real and virtual worlds and to detect the assembly errors, respectively. Meanwhile, the system can be used for desktop applications, which overcomes the original limit of HoloLens that is used for only room-scale environments and objects larger than a cube with an edge of 30 cm. Moreover, in the demonstration of decoration arrangement, the user can follow AR instruction to place decorations with the accuracy of ±1 cm in an 8 m^2 room. This indicates that the developed AR system is possible to support AR assembly assistance in room-scale environments.

Thus, the reach objectives in this study are achieved

- ✓ A basic AR assembly assistance system is developed using only the HoloLens
- ✓ The developed system can perform desktop applications by using the HoloLens-scanned point cloud to obtain 3D data of real objects.
- ✓ Present a method to evaluate misalignment between real and virtual objects
  - ➢ at real time rate (30fps) on HoloLens
  - ➢ within the accuracy of ±1 cm in the case of desktop applications

Future work will continue to improve system reliability and performance and add functions to better instruct the assembly operation. With these features, this compact system has high portability and is expected to be used in wide-ranging situations.

Chapter 3
Three-Dimensional Visualization of User’s Attention on Objects using Only Eye Tracking Glasses

3.1 Introduction

In this chapter, we focus on the training part of the AR-based field service assistance system after operations, and we study users’ 3D gaze visualization to observe user’s focus of attention using eye tracking glasses.

Eye tracking technology [46] is used and has an influence on various fields. For example, aesthetic evaluation of products is an important application. Product designers may require some subjects to observe products with different designs and simultaneously detect subjects′ eye movement to analyze their visual attention through eye tracking devices. This helps designers determine which designs are more acceptable and attractive to their target customers. In the radiology field, many studies adopted the eye tracking technology to explore how radiologists perform a diagnosis [47][48][49]. By understanding how radiologists read medical images, it is possible to prevent diagnosis errors and train novice radiologists more effectively. In addition, eye tracking technology has been applied to diversified fields, such as web page design, sports [50][51], and psychology research [52][53].

One’s visual attention is usually referred to as the visualization of one’s thought. Through analyzing the visual attention by the eye tracking technology, we may be able to understand one’s thought to help us make a decision, such as the examples of aesthetic evaluation of products and web page design. Moreover, we can catch the point and learn from one’s experiences more effectively and efficiently, such as the case of novice radiologists. These demonstrate the importance of eye tracking technology. However, most applications are used in a restricted 2D space such as a fixed screen for the fixed type of eye tracking devices. The other type of the eye tracking devices is a pair of eye tracking glasses, which contains an RGB camera and eye trackers to record user’s view and gaze information, respectively. The eye tracking glasses are mobile eye trackers and allow users to walk around, but the recorded data are scene videos with gaze fixations, which are still in 2D.

There would be more precious applications in 3D in a variety of fields. Recently, with the development of the virtual reality (VR) technology, a user who wears a VR head-mounted display can use gaze fixation on virtual objects for interaction such as object selection and operation [54]. In addition, it is possible to use the eye tracking technology in other fields such as construction. Specifically, it is important for workers to inspect construction fields. To train new construction field inspectors, it is possible to use eye tracking glasses to record the gaze information of experts and novices and analyze their gaze distinction in a practical construction field. In this case, it is better to visualize 3D gaze fixation in a 3D model of the environment instead of showing the gaze data on a video, a series of 2D images, or a panorama [55]. For a room-scale or larger environment, 3D gaze visualization can directly show the user's gaze in one model and show the spatial relationship of the scene, while showing the gaze by 2D may need a video of a few minutes or a few images to include the whole scene.

Using eye tracking glasses, we can obtain 2D gaze fixation on a recorded scene video and understand a user's saccade pathways. To display the corresponding gaze data on the 3D model of the scene for a full view of the gaze fixation, it is necessary to obtain the transformation relationship between the coordinate system of the 3D model of the scene and the 2D gaze data from the eye tracking glasses, that is, to transform the 2D gaze data into the 3D model of the scene. The use of AR (augmented reality) markers, which are attached to the environment, is a common approach to obtain the transformation matrix between the coordinate system of the 2D gaze data and that of the 3D model of the scene [56][57]. However, extra time is required to setup the experimental environment, which may disturb the subject's attention during the experiment. To avoid using the markers, some researchers rely on the image registration technique to evaluate the transformation matrix, and the 3D model of the scene is reconstructed through a scene video from an RGB camera [58], through depth maps from an RGB-D camera [59][60], or through a Light Detection and Ranging (LiDAR) scanner [61]. However, to acquire the 3D model of the scene, those methods require to scan the environment by additional devices. This may present a challenge when the subject is moving in a large-scale test environment. For example, the subject may be asked to walk around and observe the inner side of the building. Thus, it will be tedious and time-consuming to scan the entire building by other devices before or after the eye tracking experiment.

Because there is an RGB camera embedded in eye tracking glasses, and it records the scene that the user paid attention to, using only the embedded RGB camera is a possible way to reconstruct the 3D model of the scene for 3D gaze visualization. This implies that when the subject wears the eye tracking glasses and observes some environment, he/she is also recording the information for 3D reconstruction of the scene through the embedded camera. Therefore, we do not need to use extra devices to generate the scene model. In this study, we propose a methodology that uses only a pair of eye tracking glasses to achieve the visualization of 3D gaze fixation on the 3D model of the scene. We use the image-based 3D scene reconstruction method to reconstruct the 3D model of the scene and to obtain the camera position and orientation corresponding to the frame images of the scene video. We use the reconstructed camera position and orientation to determine user's line of sight and to find the 3D gaze fixation, which is the intersection of the line of sight and the scene model. With this methodology, a user's eye tracking recording can give a 3D model of the scene with his/her gaze information.

Moreover, we have to consider an issue of comparing 3D gaze fixations between multiple users. Each user is allowed to view the scene freely and there is difference in their ways of viewing the scene. If we use the method of single user's 3D gaze visualization, we have to reconstruct multiple 3D models from different users' recordings, which is time-consuming. However, those models demonstrate the same scene. Thus, in this study, we present a methodology to reconstruct only one 3D model of scene and determine different users' gaze data on that model through image registration. To sum up, the main objectives of this study include the following:

1. Use only eye tracking glasses for data collection.
2. Visualize multiple users' 3D gaze fixation more efficiently and effectively.

The preliminary results show that we can use only eye tracking glasses to reconstruct 3D model of the scene with reliable user's 3D gaze information. A room-scale experiment also shows that, for complex and large experiments, 3D gaze visualization gives a clearer picture of the eye tracking result and more reliable results than using a panorama. Moreover, for the comparison between multiple users' gaze data, due to using only a model of the scene, it shows users' gaze data in the same coordinate system and make the gaze differences between users more obvious.

In this chapter, in Section 3.2, we first review the related work on visualization of 3D gaze fixation and techniques used in this study. Section 3.3 introduces the eye tracking glasses that we use. The detailed methodology is described in Sections 3.4 and 3.5. Experiments and the effectiveness of the proposed system are discussed in Section 3.6, and Section 3.7 is the summary of this study.

3.2 Literature Review

To demonstrate user's gaze information in 3D space, displaying 3D gaze fixation on a 3D mesh model of the scene is an intuitive way, and various studies used different methods to address it. Because eye tracking glasses record user's gaze information on a scene video (i.e., in 2D space), to find the 3D position of the gaze fixation in the model of the scene, the key point is to determine a transformation relationship between the coordinate system of the 2D gaze data from the eye tracking glasses and that of the 3D model of the scene. In Fig. 3-1, Takahashi et al. [57] set AR markers in a test environment to calculate the transformation matrices and used a portable surface scanner to reconstruct the 3D model of the scene. By detecting the AR markers recorded in the scene video and finding the corresponding positions of the AR markers in the 3D model of the scene, the 3D model of the scene can be mapped into each frame image and overlap with the real scene with a corresponding pose. Thus, the 2D gaze fixation on the frame image could be displayed in the image of the 3D model of the scene. Using an inverse transform from the coordinate system of the frame image to the 3D model of the scene, the 3D gaze fixation on the 3D model can be determined.

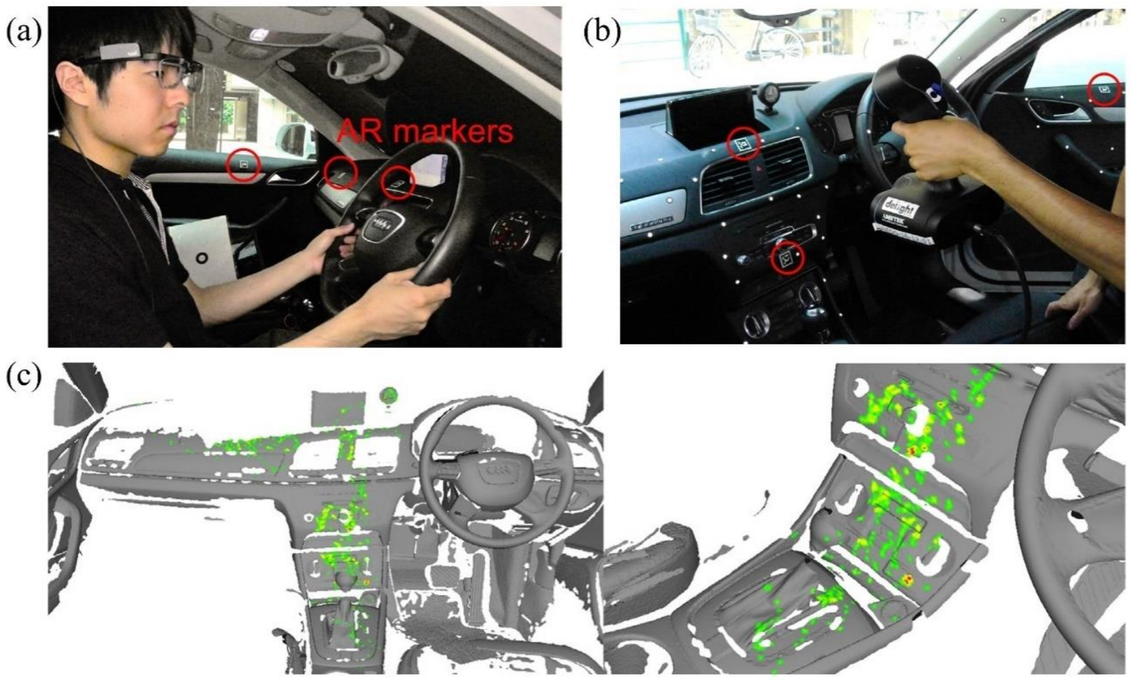


Fig. 3-1 3D gaze visualization of a car example with the use of AR markers and a portable surface scanner [57]: (a) experiment of eye tracking recording with AR markers set in the car, (b) the use of a portable surface scanner for scene reconstruction, and (c) results of 3D gaze visualization.

Using a portable surface scanner needs to set markers in the environment as feature points for the scene reconstruction, which may be tedious work. Instead, using other devices such as an RGB camera and an RGB-D camera is another choice. By extracting feature points from the images of the scene, there is no need to set physical markers in the environment. In Fig. 3-2, Paletta et al. [59] used an RGB-D camera to reconstruct a 3D model of the scene for 3D gaze visualization. Using the RGB-D camera to scan the scene from different views, point clouds of the parts of the scene could be generated. These point clouds could then be combined into the complete point cloud of the scene by estimating the position and orientation of each camera view. After the eye tracking recording, SIFT (scale-invariant feature transform) descriptors [62], as the feature descriptors of an image, for each frame image in the scene video recorded by the eye tracking glasses are calculated to determine the transformation relationship between the 2D gaze data and the 3D model of the scene using the perspective n-Point algorithm [63].

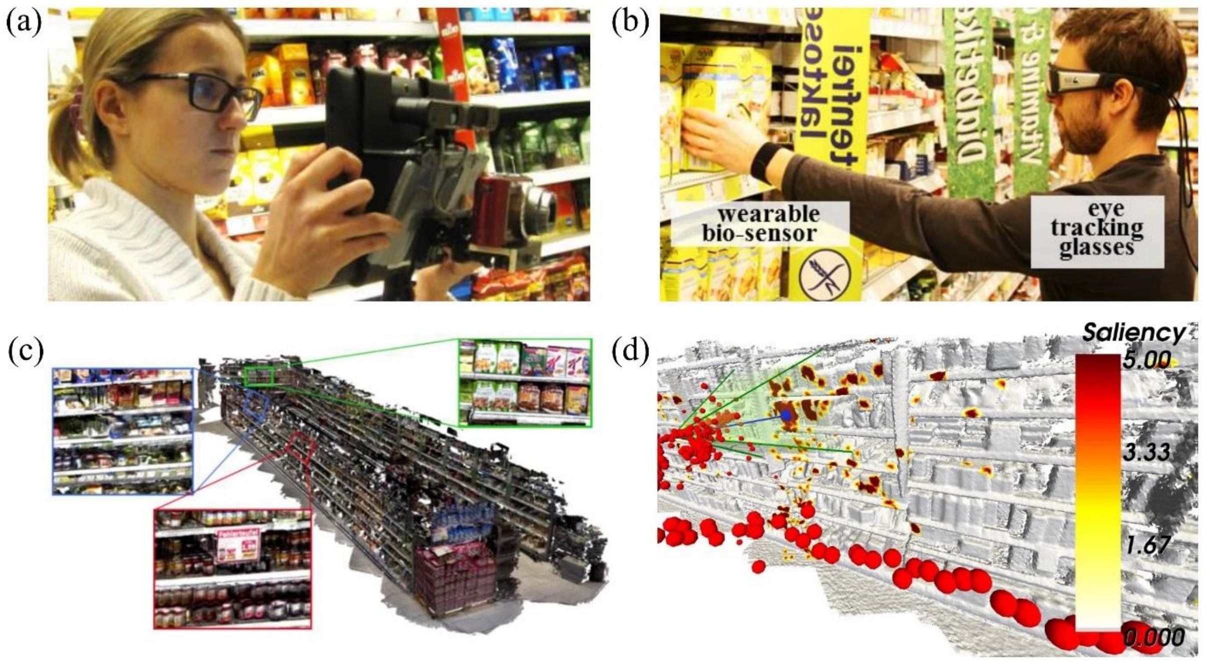


Fig. 3-2 3D gaze visualization with the use of an RGB-D camera for 3D reconstruction [59]: (a) Hardware for 3D reconstruction, (b) experiment of eye tracking recording, (c) reconstructed 3D model of the scene, and (d) results of 3D gaze visualization

Moreover, with the image-based 3D reconstruction method, the 3D model of the scene can also be constructed by using only an RGB camera. For 3D gaze visualization of this approach, Jensen et al. [58] took advantage of the SIFT descriptor matching methodology to estimate the transformation matrix of the coordinate system of the 3D model that is reconstructed from the frame images of an extra RGB camera.

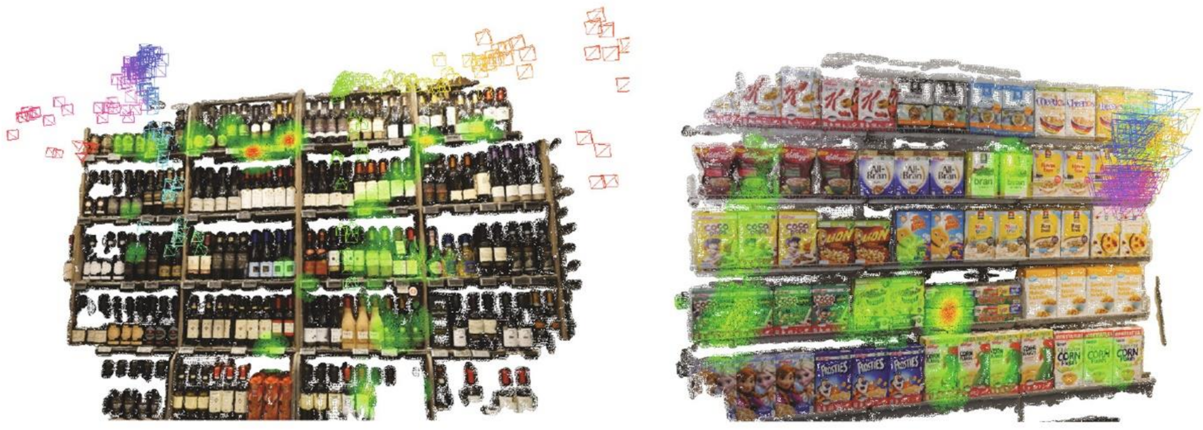


Fig. 3-3 3D gaze visualization result with the use of an extra RGB camera for 3D reconstruction [58]

Although the two studies of using the RGB-D and RGB cameras developed a markerless approach to demonstrate 3D gaze fixation on the 3D model of the scene, the experimental scenes they discussed were only shelves of goods, which can be viewed as a 2D case. Because the scenes were not complicated and sufficiently large, the advantages of demonstrating 3D gaze fixation were not adequately expressed. Furthermore, to demonstrate 3D gaze fixation, those researchers used eye tracking glasses and other devices (e.g., a surface scanner and an RGB-D camera) to record the scene video with the gaze information and reconstruct the 3D model of the scene, respectively. The proposed system only requires a pair of eye tracking glasses to both record the gaze data and to perform image-based 3D reconstruction [64][65], and no markers are placed in the environment. Additional details on the image-based 3D reconstruction will be introduced in Section 2.4.1.

3.3 Tobii Eye Tracking Glasses

In this study, we use Tobii eye tracking glasses (Tobii Pro Glasses 2) [66], which are produced by a company named Tobii Pro, to track user's eyes and to record scene videos. Fig. 3-4 shows its appearance. It is made up of two components: a wearable eye tracker (glasses) and a recording unit. The glasses have some embedded sensors, including an RGB scene camera, eye trackers, and an inertial measurement unit (IMU) to acquire various information for the subsequent analysis of user's attention. The camera records scene videos at 25 frames per second (fps), and the eye trackers record gaze data at 50 Hz. The recording unit, which is connected to the glasses via an HDMI cable, holds the battery and stores the recorded data, i.e., gaze data and scene videos, on an SD memory card. Compared with a standard eye tracking device that is fixed at a certain location, a user is able to wear the eye tracking glasses and walk around to observe surroundings. Meanwhile, simultaneously recorded gaze data and the scene video can be stored on the SD memory card for post-processing and data analysis or be transferred to a computer running Tobii Pro-provided controller software to observe eye tracking results and to perform the analysis in real time. With the provided software, we can see that the gaze data are integrated into the scene video to show where the user was looking.

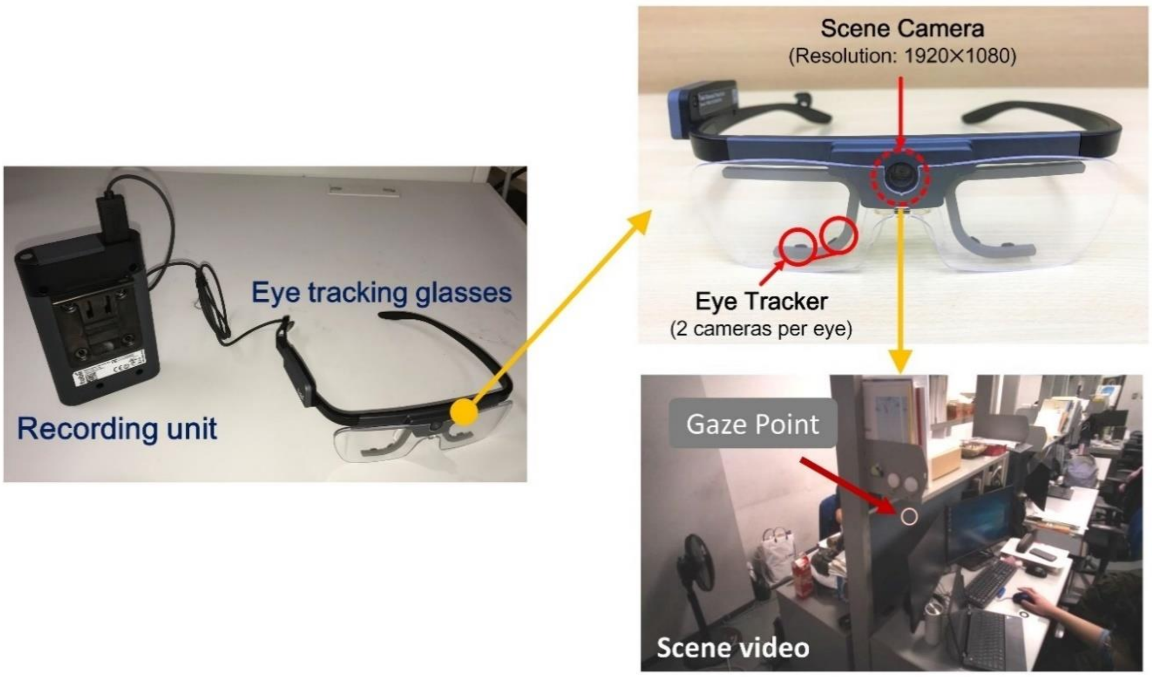


Fig. 3-4 Tobii eye tracking glasses

Furthermore, Tobii eye tracking glasses are commonly used as a device to help researchers gather users′ gaze data and then analyze users′ visual attention [58][67]. This is a reliable tool to obtain accurate gaze data. To achieve accurate eye tracking, calibration and adjustment of gaze data is necessary. Before an eye tracking recording starts, the user has to take a calibration procedure by looking at a calibration card as shown in Fig. 3-5. During the procedure, the eye trackers measure features of the user′s eyes and use them together with an internal anatomical 3D eye model to compute the gaze data. Meanwhile, during the calibration, the user is required to look at a specific target, which is also inside the viewing range of the scene camera. During this period, pixel coordinates of the user′s gaze data on the scene video are evaluated. When the calibration procedure is finished, the user can start the eye tracking recording.

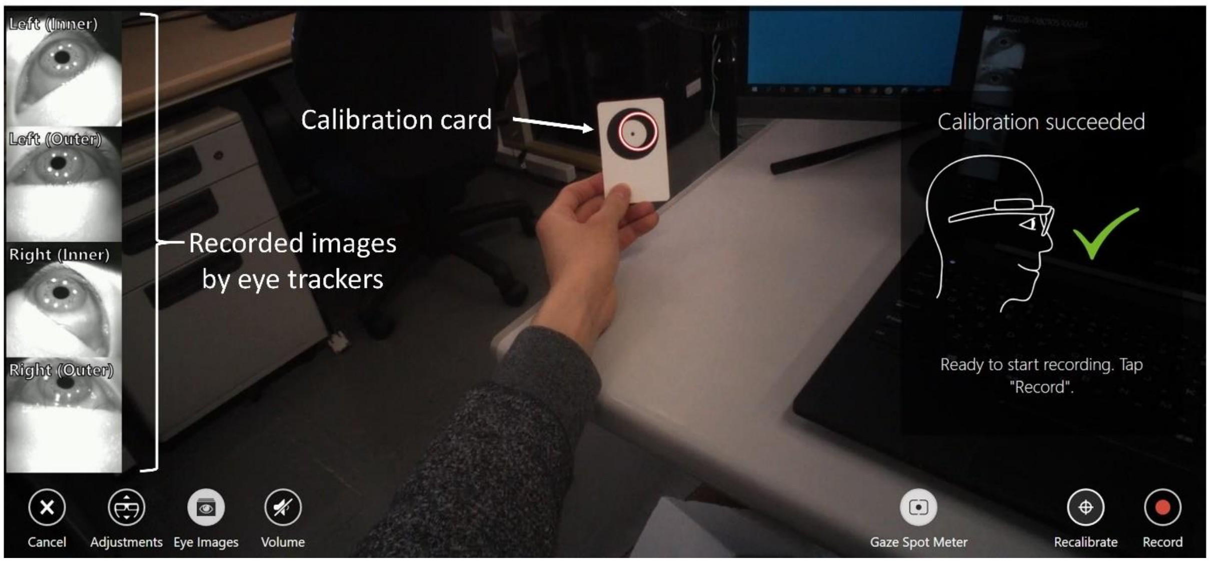


Fig. 3-5 Calibration process by looking at a calibration card

According to the eye tracker data quality report released by Tobii Pro [68], when the user who wear the Tobii eye tracking glasses observes the target at varying distances from 0.5 m to 3.0 m, the accuracy of the detected gaze angle is from 0.56° to 0.73° under optimal gaze angles (≤15°) and lighting condition of 300 lux. Thus, Tobii eye tracking glasses have good eye tracking ability for room-scale environments.

Moreover, Tobii Pro also provides Tobii Pro Lab, a commercial eye tracking software, to let the user further analyze and visualize 2D gaze data from Tobii eye tracking glasses. It can collect and analyze gaze data statistically and then generate heatmaps and saccades on images to visualize where the user was paying more attention and user's eye movement from one point of interest to another, respectively. In Section 3.6.3, we also use the Tobii Pro Lab software and then produce the 2D gaze results as a standard to compare with our 3D gaze visualization results and to analyze their differences.

3.4 Three-Dimensional Visualization of User's Gaze Data

In this section, we describe the ways that we developed to demonstrate gaze data in 3D using the data only from the eye tracking glasses. An application scenario is as follows. First, for data collection, a user wearing eye tracking walks around and pays attention to the surroundings. During that time, user's gaze data and the scene video are recorded and stored on the SD memory card. Then, the recorded data and video are input into our developed system to visualize user's gaze fixations in 3D space.

Fig. 3-6 shows the schematic diagram of the system. To achieve 3D gaze visualization, we mark the user's gaze in a 3D model of the scene generated by image-based 3D reconstruction. To determine gaze fixations in the model of the scene, we use user's line of sight that passes through user's eyes and gaze point. The intersection of the 3D model of the scene and the line of sight apparently gives the 3D gaze fixation.

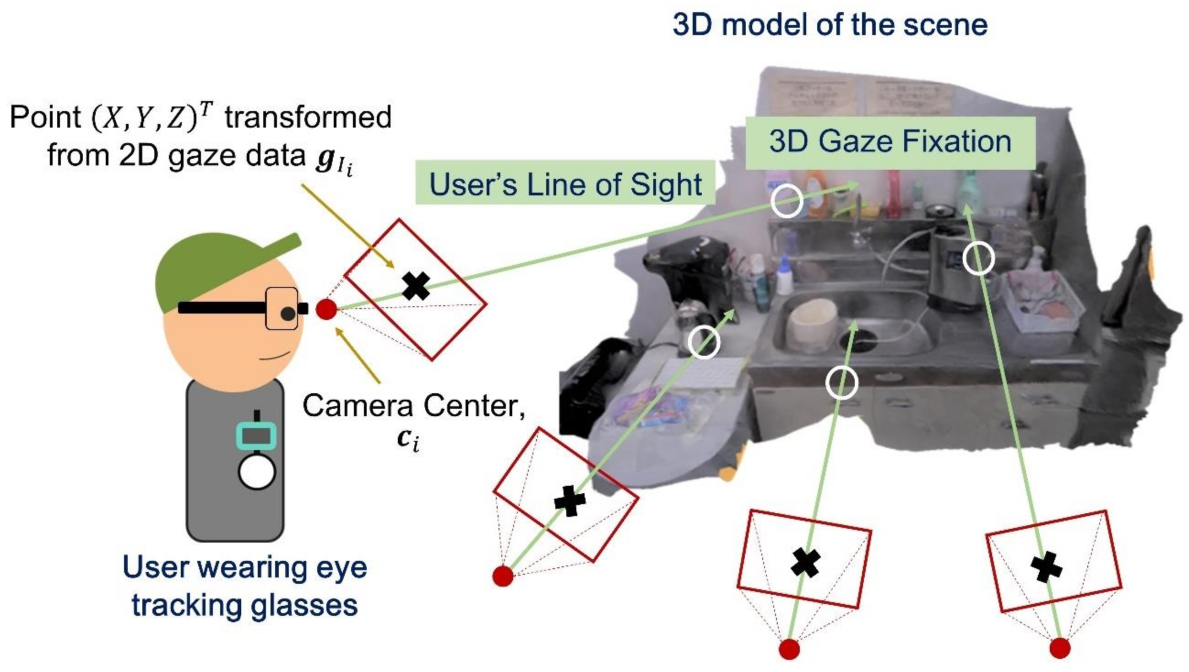

3D model of the scene
Point $(X, Y, Z)^T$ transformed from 2D gaze data $\boldsymbol{g}_{I_i}$
3D Gaze Fixation
User's Line of Sight
Camera Center, $\boldsymbol{c}_i$
User wearing eye tracking glasses

Fig. 3-6 Schematic diagram that shows 3D gaze fixation in the environmental model

Fig. 3-7 shows the proposed system's process to display the 3D gaze fixation in the 3D model of the scene. For example, a kitchen sink and its surroundings are observed by a user who wears the eye tracking glasses. The system is composed

of two parts: (1) 3D reconstruction of the environment and (2) determination of the user's line of sight and its intersection with the model of the scene.
It firstly starts from 3D reconstruction of the scene from frame images taken from the scene video using the structure from motion (SfM) method. Thus, the reconstructed 3D model then serves as a foundation to display 3D gaze fixations. Meanwhile, the position of each frame image's camera center is estimated during the 3D reconstruction stage. Next, the system determines the user's line of sight using recorded gaze data together with the position of the frame image's camera center. Thus, we are able to determine the user's gaze fixation in the 3D model of the scene through determining the intersection. Eventually, after applying such operation to all recorded gaze data, we can obtain basic 3D visualization of user's gaze data, which looks discrete and difficult to analyze user's attention. Therefore, further data processing for statistical analysis is needed. The following subsections elaborate on the process of 3D gaze visualization in detail.

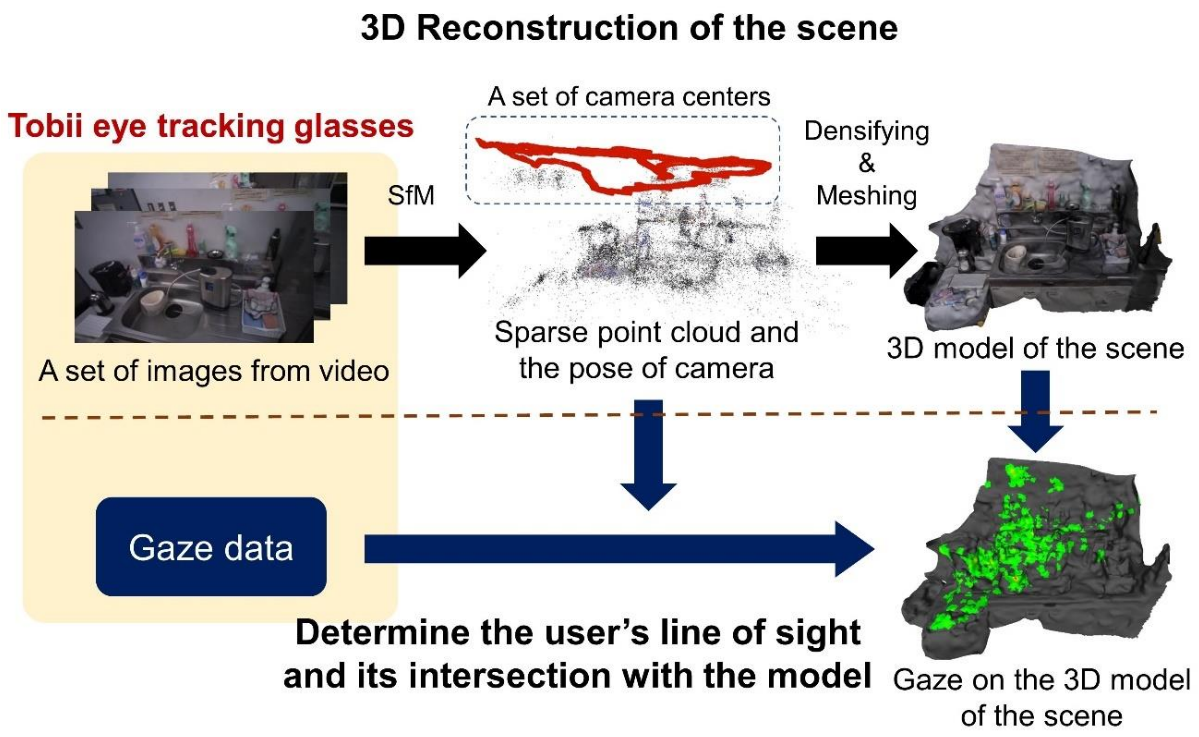

3D Reconstruction of the scene
Tobii eye tracking glasses
A set of camera centers
SfM
Densifying & Meshing
A set of images from video
Sparse point cloud and the pose of camera
3D model of the scene
Gaze data
Determine the user's line of sight and its intersection with the model
Gaze on the 3D model of the scene

Fig. 3-7 Process of 3D gaze visualization

3.4.1 3D Reconstruction of a Scene from Eye Tracking Glasses

The first stage is 3D reconstruction of the scene, producing a 3D mesh model of the scene to visualize 3D gaze data. Although there is no 3D scanner or RGB-D camera in eye tracking glasses, we can still reconstruct the 3D mesh model of the scene by the image-based 3D reconstruction method. When the user walks around in the environment, the scene camera records the target scene. Hence, we can directly make use of the frame images from the scene video for 3D reconstruction. Fig. 3-8 shows the process of image-based 3D reconstruction of the scene, which shows the same part of 3D reconstruction as shown in Fig. 3-7. For example, the kitchen sink and its surroundings are reconstructed from the scene video recorded by the eye tracking glasses. In this stage, the input is the *n* frame images extracted from the scene video, and the main output is the positions and orientations of the camera centers associated with the frame images and a 3D mesh model of the scene. To achieve the goal, this study uses the COLMAP [69][70][71][72] and the OpenMVS [73] software in sequence to perform the reconstruction.

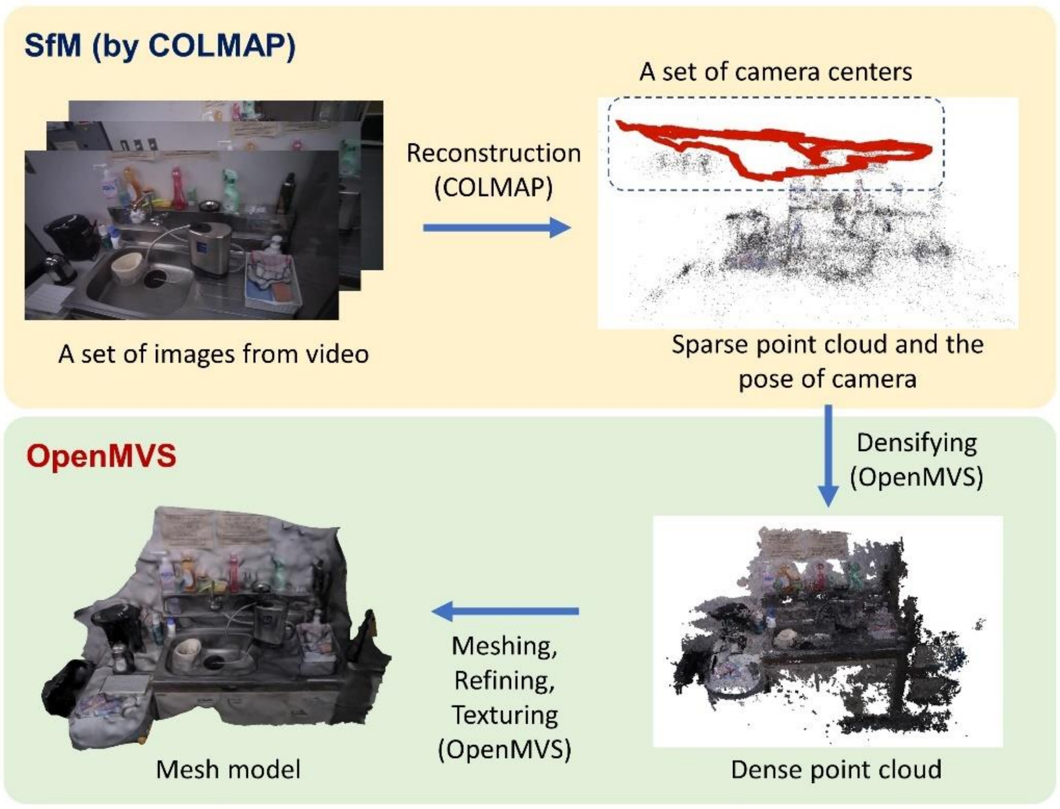


Fig. 3-8 Process of 3D reconstruction of the scene

By using the COLMAP software, the stage starts from using the SfM method [74][70], to reconstruct a sparse point cloud model of the scene and a set of camera pose $\mathcal{C} = \{\boldsymbol{c}_i\}$ (i.e., camera orientation and position) of the inputted scene video′s frame images $\mathcal{I} = \{I_i\}$. It is achieved by using the COLMAP software. According to the tests in the original paper [70], error of accumulation of camera positions is mitigated by applying bundle adjustment, and the reprojection error is averagely 0.7 pixel in their tests.

However, if we directly input all the frame images in the scene video to the system, it will be time-consuming. For example, a 1-minute video, recorded at 25 fps (the scene camera′s recording frequency), contains 1,500 frame images, and the whole 3D reconstruction process takes dozens of hours for calculation in our tests. A recommended number of images is up to several hundreds, which takes a few hours for calculation.

To reduce the computation time, we select just some frame images from the scene video for 3D reconstruction. In this application, the user has to pay attention to the environment and usually does not move the body and the head fast. Thus, in the scene video, each frame image and its neighboring frame images, captured in the same short period of time, have almost the same scene overlap. This contributes little to improve the quality of 3D scene reconstruction.

For realization, because the set of frame images taken from the video is in chronological order, it is one way to simply select frame images by downsampling the video. By starting from the first frame image I_0, we choose a frame image every n_s frame images and put them into a set of frame images $\tilde{\mathcal{I}} = \{I_{n_s i} | i = 0,1,2 \ldots\}$ especially used for the reconstruction. Through the SfM method, after feature extraction, matching, image registration, triangulation and bundle adjustment in sequence, a sparse point cloud of the scene is reconstructed from $\tilde{\mathcal{I}}$, as shown in the upper part of Fig. 3-9. Meanwhile, camera poses of $\tilde{\mathcal{I}}$ are also determined. Those extracted information are stored in a database file. The database are the tables of each image's information in $\tilde{\mathcal{I}}$, including camera intrinsic parameters (focal length, principal point, etc.), camera extrinsic parameters (orientation and location), keypoints, SIFT descriptors, and a matched image and their associated feature correspondences.

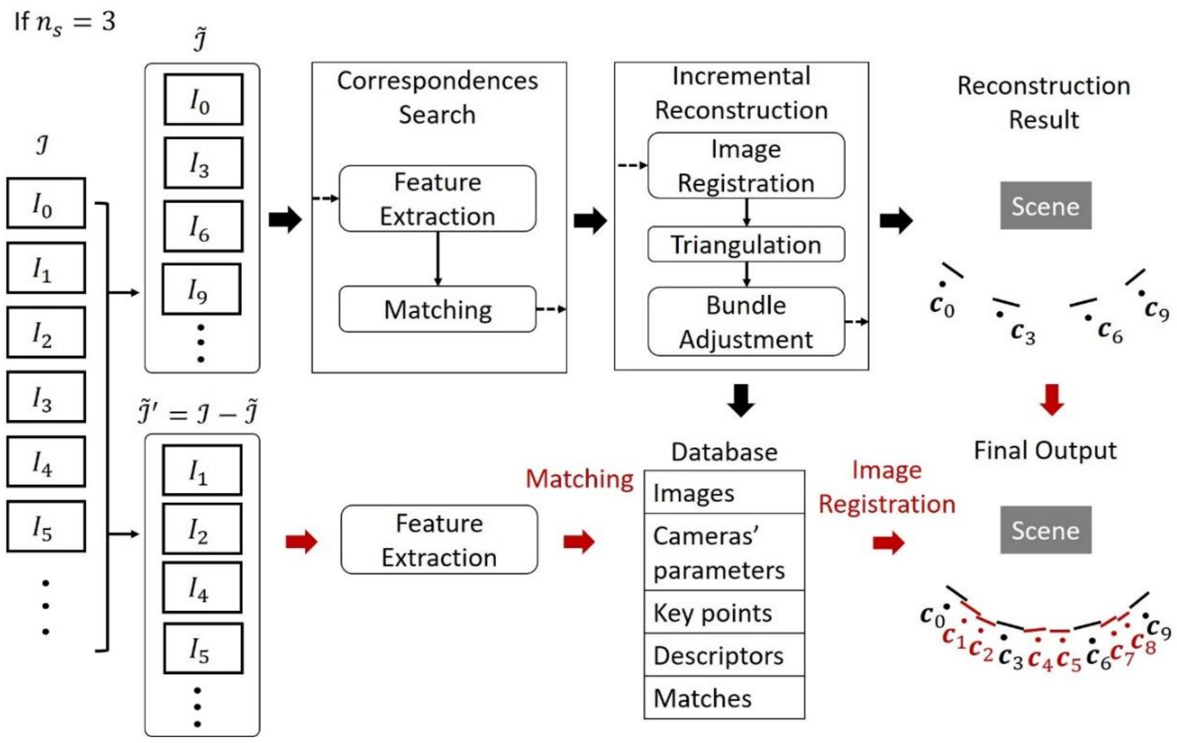


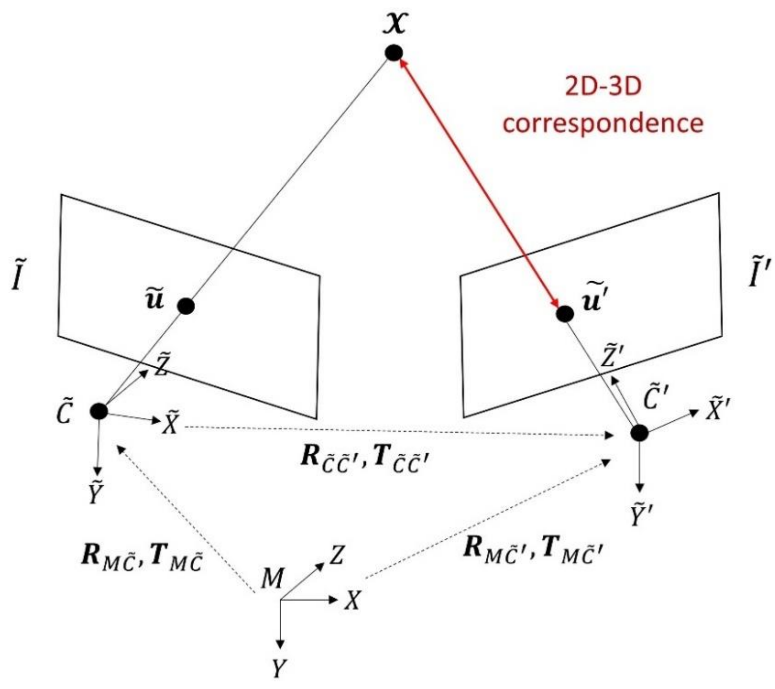
Fig. 3-9 Process of the structure from motion method with reduced frame images in the case of $n_s = 3$

For the remaining unselected frame images $\tilde{\mathcal{I}}' = \mathcal{I} - \tilde{\mathcal{I}}$, they are still needed to determine the user's line of sight. Thus, we then determine the camera pose of each image in $\tilde{\mathcal{I}}'$ under the coordinate system of the point cloud model M that is just reconstructed from $\tilde{\mathcal{I}}$, i.e., registering new images to the reconstructed model. For each image in $\tilde{\mathcal{I}}'$, we first extract feature points and search for an overlapping image in the database (from $\tilde{\mathcal{I}}$) that sees the same scene part. Then, through two-view geometry [74], the camera poses of the images in $\tilde{\mathcal{I}}'$ can be estimated using feature correspondences to triangulated 3D points (2D-3D correspondences) in already registered images, belonging to the database generated from $\tilde{\mathcal{I}}$. This is illustrated in Fig. 3-10. Basically, given an image $\tilde{I}'$ in $\tilde{\mathcal{I}}'$ and its corresponding paired image $\tilde{I}$ in $\tilde{\mathcal{I}}$ determined in the matching process, an essential matrix relating the pair of views can be used to determine the camera coordinate system $\tilde{C}'$ with respect to $\tilde{C}$, the camera coordinate system relating to $\tilde{I}$. In addition, $\tilde{\boldsymbol{u}}$ and $\tilde{\boldsymbol{u}}'$ are one pair of point correspondences between $\tilde{I}$ and $\tilde{I}'$, and they are the projection of a 3D point $\boldsymbol{X}$

in different views. The essential matrix can be estimated linearly using 8 or more point correspondences and then decomposed to give relative camera orientation $\boldsymbol{R}_{\tilde{C}\tilde{C}'}$ and the direction of camera translation $\boldsymbol{T}_{\tilde{C}\tilde{C}'}$. The magnitude of the translation $\|\boldsymbol{T}_{\tilde{C}\tilde{C}'}\|$ can be determined using the projection $\tilde{\boldsymbol{u}}'$ in the image $\tilde{I}'$ of the single known 3D point $\boldsymbol{\mathcal{X}}$, i.e., a point that has already been reconstructed from the images in $\tilde{\mathcal{I}}$ and saved in the database. The transformation relationship between $\tilde{C}$ and $\tilde{C}'$ can be written in the homogeneous coordinates as:

$$\begin{bmatrix} \tilde{X}' \\ \tilde{Y}' \\ \tilde{Z}' \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_{\tilde{C}\tilde{C}'} & \boldsymbol{T}_{\tilde{C}\tilde{C}'} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \tilde{X} \\ \tilde{Y} \\ \tilde{Z} \\ 1 \end{bmatrix} \tag{10}$$

where $\boldsymbol{R}_{\tilde{C}\tilde{C}'}$ is a 3×3 rotation matrix, and $\boldsymbol{T}_{\tilde{C}\tilde{C}'}$ is a 3×1 translation vector. Equation (10) shows the rigid body transformation that relates points $[\tilde{X} \quad \tilde{Y} \quad \tilde{Z}]^T$ in the camera coordinate system $\tilde{C}$ to points $[\tilde{X}' \quad \tilde{Y}' \quad \tilde{Z}']^T$ in the camera coordinate system $\tilde{C}'$.



$\boldsymbol{\mathcal{X}}$
2D-3D correspondence
$\tilde{I}$
$\tilde{\boldsymbol{u}}$
$\tilde{I}'$
$\tilde{\boldsymbol{u}}'$
$\tilde{Z}$
$\tilde{C}$
$\tilde{X}$
$\tilde{Y}$
$\tilde{Z}'$
$\tilde{C}'$
$\tilde{X}'$
$\tilde{Y}'$
$\boldsymbol{R}_{\tilde{C}\tilde{C}'}, \boldsymbol{T}_{\tilde{C}\tilde{C}'}$
$\boldsymbol{R}_{M\tilde{C}}, \boldsymbol{T}_{M\tilde{C}}$
$\boldsymbol{R}_{M\tilde{C}'}, \boldsymbol{T}_{M\tilde{C}'}$
M
Z
X
Y

Fig. 3-10 Estimation of the newly registered image's camera pose

Furthermore, because $\tilde{I}$ is one of the images used for 3D reconstruction, the corresponding camera coordinate system $\tilde{C}$ relative to the coordinate system of the reconstructed model M, i.e., the world coordinate system, has been determined and saved in the database. Similarly, the transformation relationship can be written as:

$$\begin{bmatrix} \tilde{X} \\ \tilde{Y} \\ \tilde{Z} \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_{M\tilde{C}} & \boldsymbol{T}_{M\tilde{C}} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{11}$$

where $\boldsymbol{R}_{M\tilde{C}}$ and $\boldsymbol{T}_{M\tilde{C}}$ are a 3x3 rotation matrix and a 3x1 translation vector, respectively. They relate points $[X\ Y\ Z]^T$ in M to points $[\tilde{X}\ \tilde{Y}\ \tilde{Z}]^T$ in $\tilde{C}$**.** By integrating Eq. (10) and (11), we can then determine the camera coordinate system $\tilde{C}'$ of the newly registered image $\tilde{I}'$ relative to M as follows:

$$\begin{bmatrix} \tilde{X}' \\ \tilde{Y}' \\ \tilde{Z}' \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_{\tilde{C}\tilde{C}'} & \boldsymbol{T}_{\tilde{C}\tilde{C}'} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{R}_{M\tilde{C}} & \boldsymbol{T}_{M\tilde{C}} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_{M\tilde{C}'} & \boldsymbol{T}_{M\tilde{C}'} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{12}$$

where $\boldsymbol{R}_{M\tilde{C}'}$ and $\boldsymbol{T}_{M\tilde{C}'}$ are the rotation matrix and the translation vector from M to $\tilde{C}'$ and describe the camera pose of $\tilde{C}'$ in M. Therefore, by substituting $[0\quad 0\quad 0\quad 1]^T$ into $[\tilde{X}'\quad \tilde{Y}'\quad \tilde{Z}'\quad 1]^T$ in Eq. (12), it gives $-\boldsymbol{R}_{M\tilde{C}'}^{-1}\boldsymbol{T}_{M\tilde{C}'}$ as the camera center of the image $\tilde{I}'$ that is the position of the origin of $\tilde{C}'$ in M. After registering all the images in $\tilde{\mathcal{I}}'$ to M, we then have the sparse point cloud model M of the scene as well as the camera pose information of all the frame images $\mathcal{I}$ from the input scene video.

The next stage is to produce a mesh model of the scene from the reconstructed sparse point cloud. Using the mesh model of the scene is beneficial to determining the position of the 3D gaze fixation in the model and visualizing the 3D gaze fixation. In terms of visual effect, colorizing triangular meshes to mark the location of the user's attention is more obvious than colorizing points. This process is held by using the OpenMVS software. The input is the sparse point cloud of the scene and the camera pose information of the frame images $\tilde{\mathcal{I}}$, i.e., the output of SfM from the previous stage, as shown in Fig. 3-11(b).

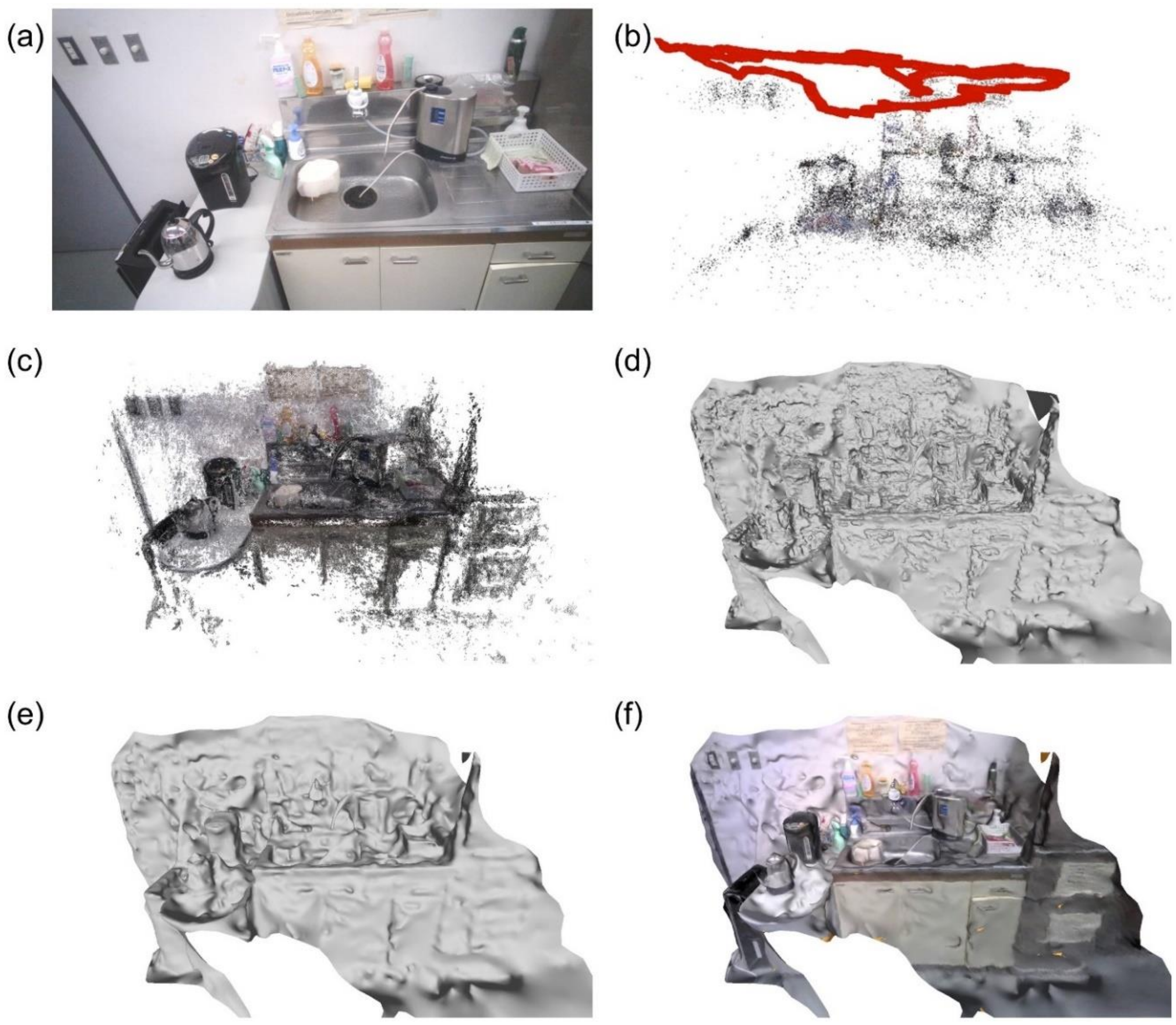


Fig. 3-11 Process of 3D surface reconstruction based on the output of SfM: (a) target scene, (b) outputted sparse point cloud from SfM, (c) dense point cloud, (d) rough mesh reconstruction, (e) refined mesh model, and (f) final mesh model with texture

The process for reconstructing the mesh model in OpenMVS contains dense point cloud reconstruction, mesh reconstruction, mesh refinement, and mesh texturing performed in this sequence. First, dense point cloud reconstruction is applied to obtain a complete and accurate point cloud as possible, generating a dense point cloud model of the scene, as shown in Fig. 3-11(c). Since the reconstruction by using the SfM method, a set of 3D points corresponding to the features extracted from the images, is usually sparse, it is a necessity to obtain a dense representation of the target scene before the meshing process. This problem can be handled by the multi-view stereo method [75]. The multi-view stereo algorithm is a common solution in photogrammetry applications for the dense reconstruction of a static scene from images alone. In our application, it can take camera location, camera orientation, and other information from SfM to make a dense 3D point cloud model of the scene. OpenMVS estimates a depth map for each view (image) based on the PatchMatch algorithm [76]. By considering the depth map as a 2D array of 3D points, multiple depth maps can be merged and become a highly detailed 3D point cloud of the scene.

Next, mesh reconstruction [77] is applied to the dense point cloud to generate 3D triangular mesh surfaces, as shown in Fig. 3-11(d). In Fig. 3-11(e), the reconstructed mesh model is further refined with a variational multi-view stereo vision approach [78][79] including photo-consistency measurement between the images and surface regularization for the reprojection error minimization and the improvement of smoothness while preserving the details of the 3D surface, respectively. Eventually, mesh texturing [80] is implemented to add color information to the reconstructed model from the images, as shown in Fig. 3-11(f). Thus, with the processes, we can acquire a colored 3D surface reconstruction of the scene from the recorded scene video and use the mesh model to show user′s gaze data. Moreover, during the reconstruction, the camera pose of each frame image is generated and can be used to determine the user's line of sight and the 3D gaze fixation on the reconstructed model.

3.4.2 Computation of Corresponding Gaze Data for Each Frame

Next, to determine the user's line of sight for each frame image I_i, we need the gaze position in the frame image, which gives information about the user's line of sight. However, the scene video and the original gaze data $G = \{\boldsymbol{g}_0, \boldsymbol{g}_1, \dots, \boldsymbol{g}_j, \dots\}$ are recorded by the scene camera and the eye trackers, respectively, at different sampling frequencies, where $\boldsymbol{g}_j = [\mu_{\boldsymbol{g}_j} \ \nu_{\boldsymbol{g}_j}]^T$ is the pixel coordinates of 2D gaze position on the image. Therefore, a prerequisite is to synchronize the video and the gaze data, and then compute the corresponding gaze data $\boldsymbol{g}_{I_i} = [\mu_{I_i} \ \nu_{I_i}]^T$ for each frame image I_i. This can be achieved by comparing the timestamps of the gaze data and of the frame image and performing linear interpolation to estimate $\boldsymbol{g}_{I_i}$.

As shown in Fig. 3-12, the timestamp t_i of each frame image is indicated by the presentation timestamp, which determines when the frame should be presented in the video, and extracted by the FFmpeg software [81]. The timestamp of the first frame image t_0 in the video is zero. However, the timestamp of the gaze data $\hat{s}_j$ is recorded based on the system time, which indicates the amount of time that has passed since the system of the eye tracking glasses was booted. Thus, the timestamp of the gaze data does not start from zero, i.e., $\hat{s}_0 \neq 0$. To synchronize the gaze data with the video frames, the first step is to translate the timestamp of the frame image t_i into the system-based timestamp $\hat{t}_i$ using synchronization information t_{sync}, which is also recorded by the eye tracking glasses and indicates the offset between the timestamp of the gaze data and of the frame image, as shown in Eq. (13):

$$\hat{t}_i = t_i + t_{sync} \tag{13}$$

After the translation of the timestamp of the video frame, a linear interpolation method is used to determine the corresponding pixel coordinate of the gaze data $\boldsymbol{g}_{I_i}$ in each frame I_i. Each recorded timestamp of the gaze data $\hat{s}_j$ corresponds to one gaze data $\boldsymbol{g}_j$. Therefore, the gaze data $\boldsymbol{g}_{I_i}$ for each frame I_i can be estimated by determining two neighboring timestamps of the gaze data, $\hat{s}_j$ and $\hat{s}_{j+1}$, that are closest to $\hat{t}_i$ and including it followed by performing the interpolation using Eq. (14).

$$\boldsymbol{g}_{I_i} = \begin{bmatrix} \mu_{I_i} \\ \nu_{I_i} \end{bmatrix} = \begin{bmatrix} \mu_{\boldsymbol{g}_j} \\ \nu_{\boldsymbol{g}_j} \end{bmatrix} + \frac{\hat{t}_i - \hat{s}_j}{\hat{s}_{j+1} - \hat{s}_j} \times \begin{bmatrix} \mu_{\boldsymbol{g}_{j+1}} - \mu_{\boldsymbol{g}_j} \\ \nu_{\boldsymbol{g}_{j+1}} - \nu_{\boldsymbol{g}_j} \end{bmatrix} \tag{14}$$

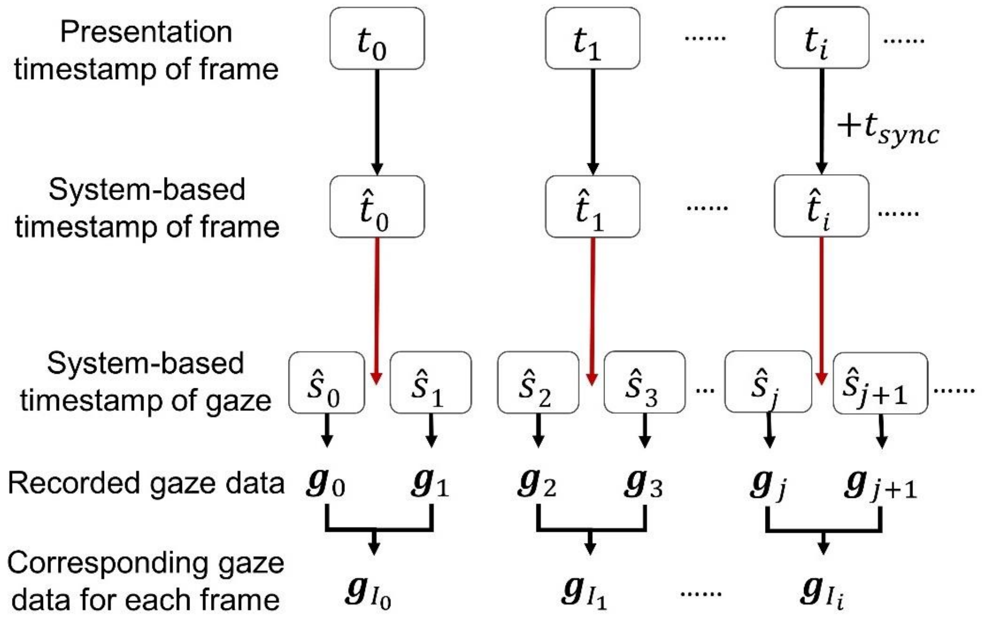


Fig. 3-12 Synchronization of the timestamp of the frame and gaze data

3.4.3 Determination of 3D Gaze Fixation

After the 3D reconstruction of the scene, the next step is to determine 3D gaze fixation on the reconstructed model M. Fig. 3-13 shows the relationship between recorded 2D gaze data $\boldsymbol{g}_{I_i}$ in the pixel coordinate and its corresponding 3D gaze fixation $\mathbf{X}_{I_i}$ on the 3D scene.

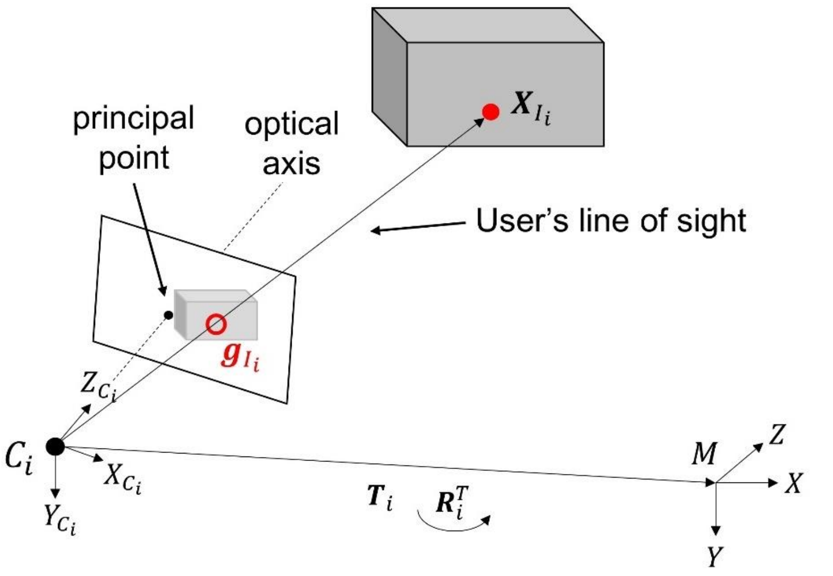


Fig. 3-13 Relationship between 2D gaze data and corresponding 3D gaze fixation

Under the pinhole projection model, gaze data $\boldsymbol{g}_{I_i}$ is the projected pixel position of its corresponding scene point $\mathbf{X}_{I_i}$, and the camera center $\boldsymbol{c}_i$ (origin of C_i), $\boldsymbol{g}_{I_i}$ in the camera image plane, and $\mathbf{X}_{I_i}$ are on the same line, which is referred to as user's line of sight in three-dimensional space. Hence, 3D gaze fixation on the reconstructed model can be recovered by backward projection from $\boldsymbol{g}_{I_i}$ to $\mathbf{X}_{I_i}$. However, the reconstructed $\mathbf{X}_{I_i}$ can only be recovered up to a one-parameter ambiguity corresponding to its distance from the camera center.

To solve the ambiguity of the distance, under the assumption that the camera center $\boldsymbol{c}_i$, the gaze point on the camera image plane, and $\mathbf{X}_{I_i}$ are all on the user's line of sight, 3D gaze fixation on the reconstructed model can be determined by the intersection of the user's line of sight and the reconstructed model without considering the distance ambiguity. Therefore, for each view, the determination of its corresponding 3D gaze fixation on the model is composed of two steps: (1) the determination of user's line of sight and (2) the determination of the intersection of the user's line of sight and the model that is the target position of 3D gaze fixation.

First, for each frame image (view), user's line of sight can be obtained by correlating the camera center $\boldsymbol{c}_i$ and the corresponding gaze point in the coordinate system of the model. The 3D position of $\boldsymbol{c}_i$ in the coordinate system of the reconstructed model M has been obtained simultaneously during the reconstruction of the 3D scene by the structure from motion algorithm (Section 3.4.1). The 3D position of the gaze point can be reconstructed from 2D gaze data $\boldsymbol{g}_{I_i}$ by using the familiar 3D to 2D transformation from the coordinate system of the model to the pixel coordinate in the image. Using homogeneous coordinates, a 3D gaze point $[X_i \; Y_i \; Z_i]^T$ in the coordinate system of the model that is related to the pixel position of gaze data $\boldsymbol{g}_{I_i} = [\mu_{I_i} \; \nu_{I_i}]^T$ may be defined up to scale by using Eq. (15):

$$s\begin{bmatrix}\mu_{I_i}\\ \nu_{I_i}\\ 1\end{bmatrix} = \boldsymbol{K}[\boldsymbol{R}_i \quad \boldsymbol{T}_i]\begin{bmatrix}X_i\\ Y_i\\ Z_i\\ 1\end{bmatrix} = \boldsymbol{K}\left(\boldsymbol{R}_i\begin{bmatrix}X_i\\ Y_i\\ Z_i\end{bmatrix} + \boldsymbol{T}_i\right) \tag{15}$$

where s is a scale factor, $\boldsymbol{R}_i$ is a 3 $\times$ 3 rotation matrix that represents camera orientation, and $\boldsymbol{T}_i$ is a vector with 3 elements that represents camera translation. Both $\boldsymbol{R}_i$ and $\boldsymbol{T}_i$ have been determined during the 3D reconstruction process and describe the camera pose. $\boldsymbol{K}$ is known as the camera intrinsic matrix and has the form of:

$$\boldsymbol{K} = \begin{bmatrix} f_x & \gamma & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \tag{16}$$

where f_x and f_y are the focal length, which is expressed in units of the pixel dimension, γ is the skew parameter, and $[p_x \; p_y]^T$ is the pixel position of the principal point. The principal point is the intersection of the optical axis (principal axis), which is a line through the camera center orthogonal to the camera image plane, with the frame image plane. It is an indication of the camera center in the image. Moreover, pixels are usually assumed to be square and, in that case, $f_x = f_y = f$ and $\gamma = 0$. Thus, we can rewrite $\boldsymbol{K}$ as:

$$\boldsymbol{K} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \tag{17}$$

In Eq. (15), all the parameters are known except the undetermined 3D homogeneous gaze point. Hence, from Eq. (15), the 3D gaze point can be obtained by the following matrix equation:

$$\begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} = (\boldsymbol{K}\boldsymbol{R}_i)^{-1} \left(s \begin{bmatrix} \mu_{I_i} \\ \nu_{I_i} \\ 1 \end{bmatrix} - \boldsymbol{K}\boldsymbol{T}_i \right) \tag{18}$$

Given recorded gazed data $\boldsymbol{g}_{I_i}$, Eq. (18) gives a 3D gaze point using the projection transformation matrix. Moreover, scale factor s determines only how far away the 3D gaze point is from the camera center, i.e., s is independent of the direction of the 3D gaze point relative to the camera. Thus, we can simply set $s = 1$. User's line of sight can then be determined by a line that passes through both the camera center and the determined 3D gaze point.

The final stage is the determination of 3D gaze fixation in the scene model through the intersection of the user's line of sight and the model. Because the surface of the reconstructed mesh model is represented by a set of triangles that are formed from the dense point cloud, the goal can be considered to find out which triangle(s) the user's line of sight passes through. Triangles that are intersected by the user's line of sight are considered as potential 3D gaze fixation, and the triangle that is closest to the camera center is then considered as the target 3D gaze fixation. To efficiently search for the intersected triangles in a model with a large number of 3D triangles, we use the computational geometry algorithms library (CGAL) software. It can construct an axis-aligned bounding box (AABB) tree data structure to store the set of 3D triangles hierarchically and use the AABB tree data structure to speed up intersection queries [82]. Through this way, we input the reconstructed 3D mech model M into the AABB tree data structure to find the triangles intersected by user's line of sight, which is a ray starting from the position of the camera center. From all the intersected triangles, we choose a triangle τ_i that is closest to the camera center corresponding to the 2D gaze data $\boldsymbol{g}_{I_i}$ in the frame image I_i as the 3D gaze visualization on the model M.

3.4.4 Gaze Visualization via the Heatmap

We compute τ_i for all the gaze fixations $\{\boldsymbol{g}_{I_i}\}$. As the triangle is intersected by more than one intersection line, we count the number of intersections P_i of τ_i. P_i indicates how many times the user's lines of sight intersect a triangle. After finding all intersected triangles, they are marked by different colors depending on the number of intersections P_i. For instance, in Fig. 3-14(a), if P_i is large, its color is red and vice versa. However, by showing only intersected triangles, the result looks discrete and difficult to analyze by the user. Thus, it is better to visualize the information of 3D gaze fixation on the mesh model as a heatmap; a diffusion filter is used to smooth the data, as shown in Eq. (19). It distributes the gaze fixation count and highlights where the user was looking. The diffusion filter has an effect that is equivalent to the Gaussian filter, which has its basis in the human visual perception system [83][84][85]. Equation (19) updates the fixation count P_i of a certain triangle τ_i in M based on the fixation count of the neighboring triangles N_i, as shown in Eq. (20):

$$P_i^{'} = P_i + \lambda \Delta P_i \tag{19}$$

where λ is the diffusion rate that is similar to the thermal diffusivity in the heat equation and controls the transfer rate of the fixation count, and

$$\Delta P_i = \frac{1}{|N_i|}\left(\sum_{j \epsilon N_i} P_j\right) - P_i \tag{20}$$

where N_i is a set of neighboring triangles to τ_i and $|N_i|$ is its number. By applying Eqs. (19) and (20) to all triangles, one iteration was formed. By performing more iterations, the gaze distribution result becomes smoother and identifies the region to which the user was paying attention with a red color. Fig. 3-14 shows the heatmap gaze distribution results for different number of iterations.

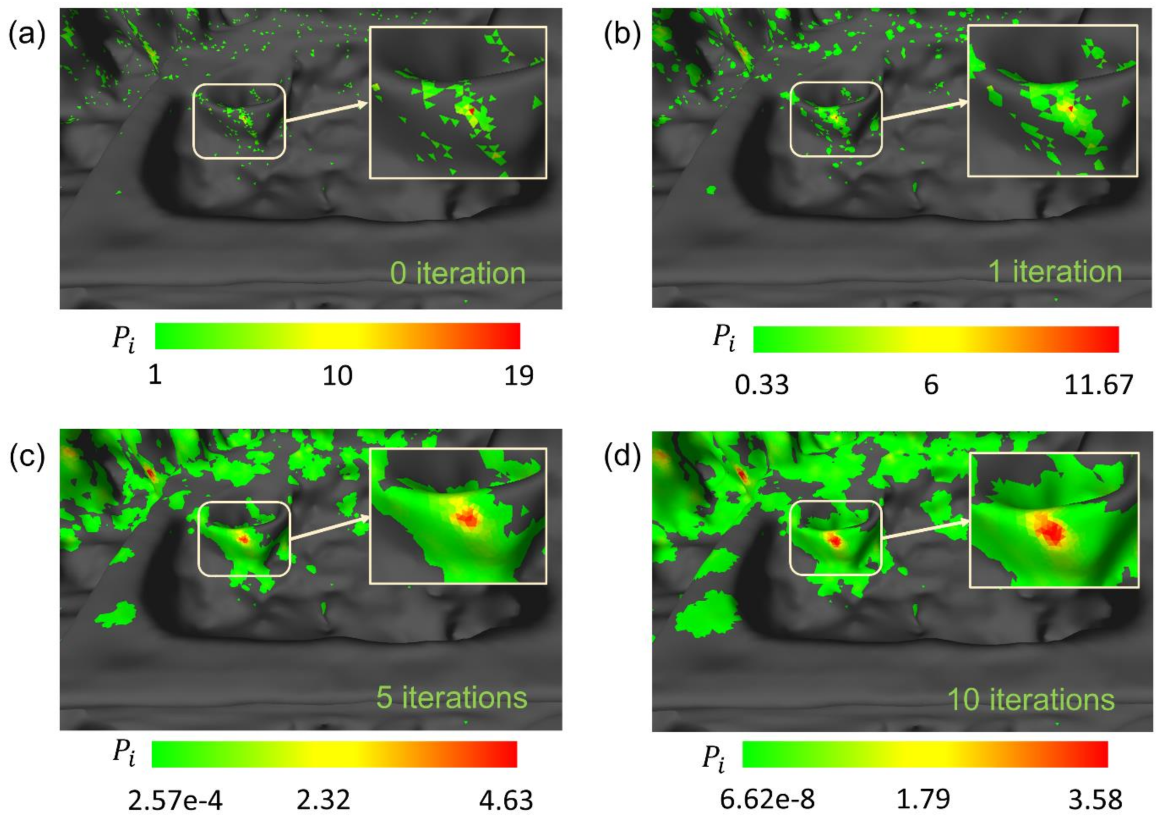


Fig. 3-14 Heatmap result with λ = 0.5 and different number of iterations

3.5 Multiple Users′ 3D Gaze Visualization Based on the Same Model

Using the approaches described in Section 3.4, the system can generate a user's 3D gaze fixations and show them on the 3D model that was reconstructed from the user’s recording. According to this idea, for multiple users who observe the same environment, the system will reconstruct a 3D model of the scene for each user from their own recording video, and then demonstrate users’ 3D gaze fixation on their own 3D model of the scene. However, it means that the system will reconstruct multiple 3D models that represent the same environment, and this will be quite time-consuming to generate multiple users’ 3D gaze visualization. An appropriate way is to reconstruct only one 3D model of the scene and then register each user’s gaze data into the model for 3D gaze visualization.

Multiple users who observe the same scene mean a group of users who are restricted to observe the same environment, but each user is able to observe the scene in his/her own way. Users can observe the scene from different views, paths, and speeds. Fig. 3-15 shows a schematic plot of multiple users’ 3D gaze visualization for three users A, B, and C. The three users observe the same scene, and we focus on the realization of their 3D gaze results by sharing the same 3D model of the scene. For the workers’ training, our proposed method can more efficiently compare the difference of 3D gaze data between workers who work in the same environment.

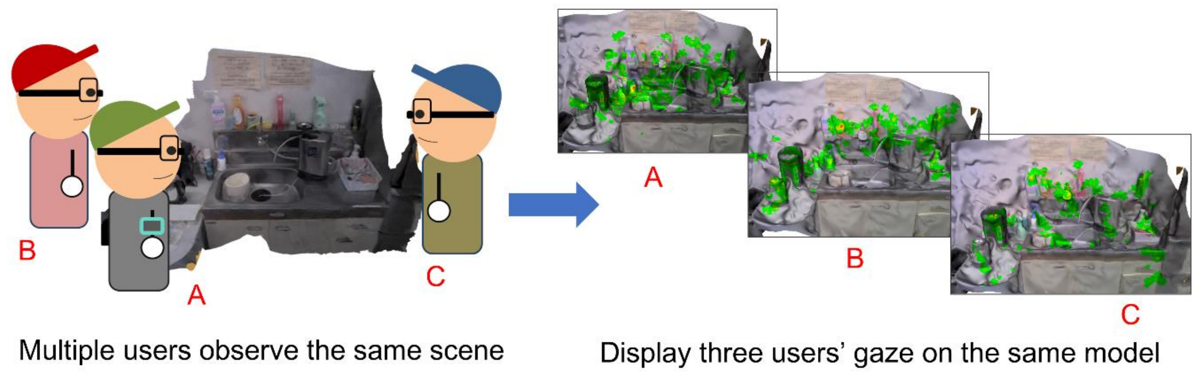

B
A
C
A
B
C
Multiple users observe the same scene
Display three users’ gaze on the same model

Fig. 3-15 A schematic plot of multiple users′ 3D gaze visualization

Fig. 3-16 shows the original method introduced in Sec. 3.4 to generate three users' 3D gaze results. Each 3D gaze visualization is reconstructed from each user's eye tracking recording. However, because the users observe the same scene, the reconstructed 3D models represent the same environment. Thus, it is not necessary to generate multiple scene models, and only one scene model is possible to display all users' 3D gaze fixation. An illustration of this idea is shown in Fig. 3-17. Reconstructing only a 3D model of the scene M_A from User A's scene video, model M_A can be shared to visualize other users' 3D gaze fixation. Without reconstructing multiple 3D models (M_B and M_C), it is expected that a large amount of time can be saved.

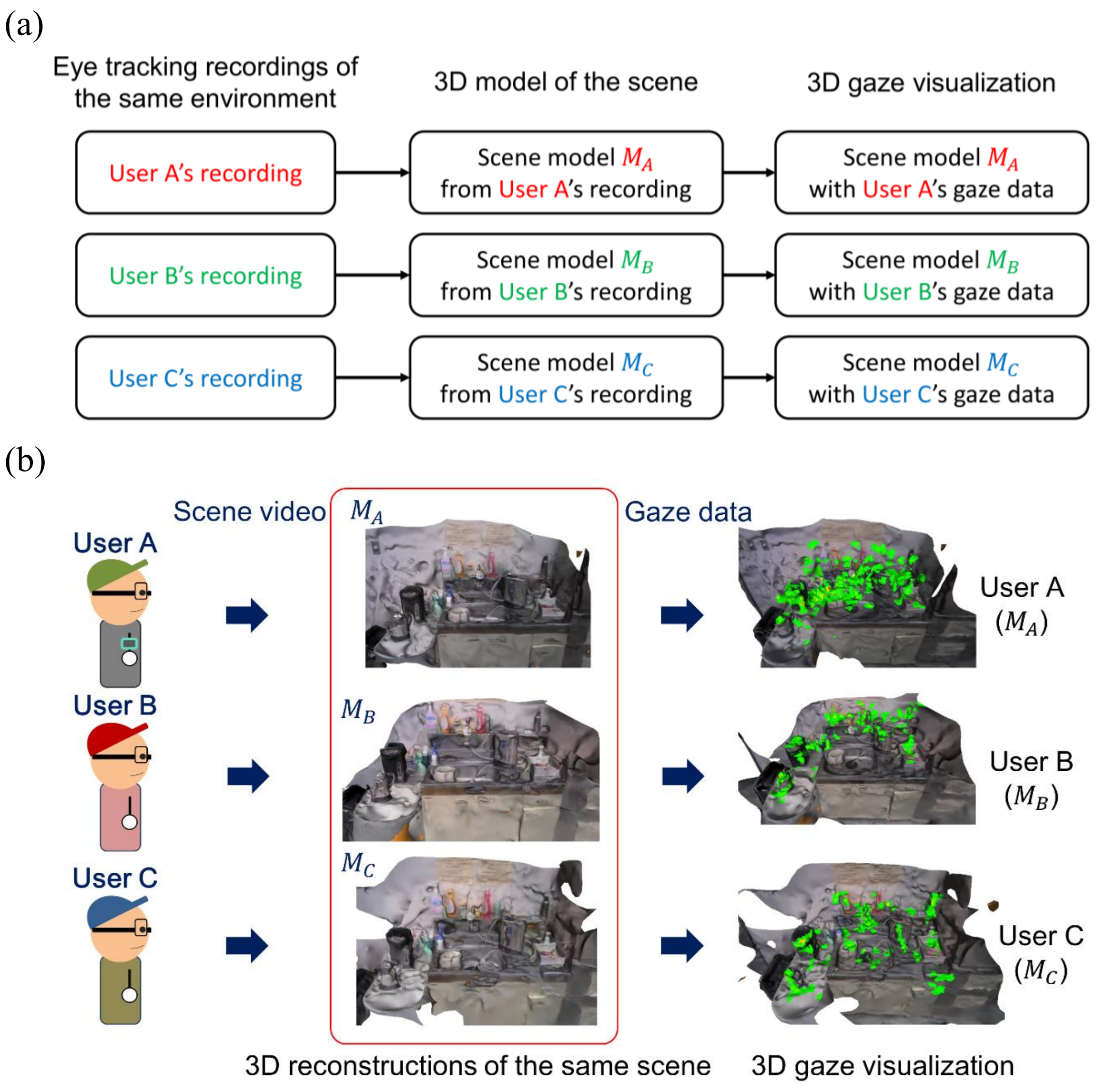
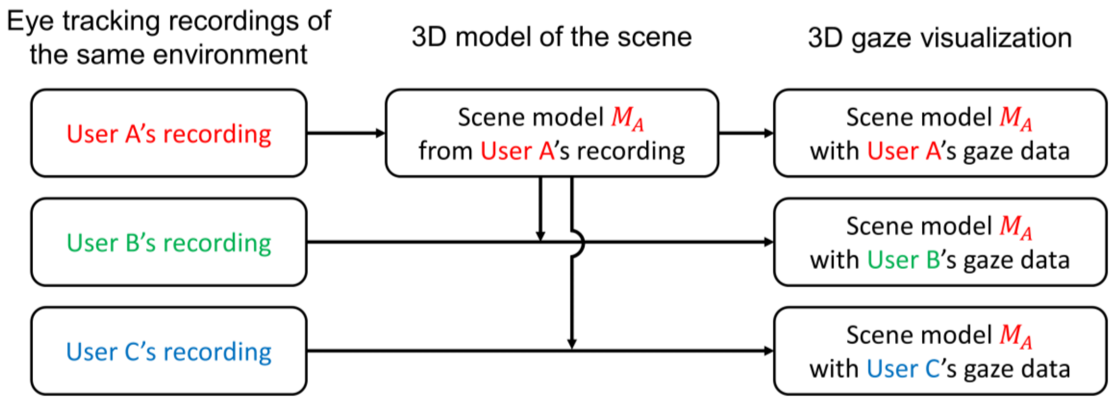

(a)
Eye tracking recordings of the same environment
3D model of the scene
3D gaze visualization
User A's recording
Scene model M_A from User A's recording
Scene model M_A with User A's gaze data
User B's recording
Scene model M_B from User B's recording
Scene model M_B with User B's gaze data
User C's recording
Scene model M_C from User C's recording
Scene model M_C with User C's gaze data
(b)
Scene video
Gaze data
User A
User B
User C
M_A
M_B
M_C
User A (M_A)
User B (M_B)
User C (M_C)
3D reconstructions of the same scene
3D gaze visualization

Fig. 3-16 Flowcharts of visualizing each user's gaze data on the model reconstructed from user's own recording: (a) text explanation and (b) a visualized plot

(a)


Eye tracking recordings of the same environment
3D model of the scene
3D gaze visualization
User A's recording
User B's recording
User C's recording
Scene model M_A from User A's recording
Scene model M_A with User A's gaze data
Scene model M_A with User B's gaze data
Scene model M_A with User C's gaze data

(b)

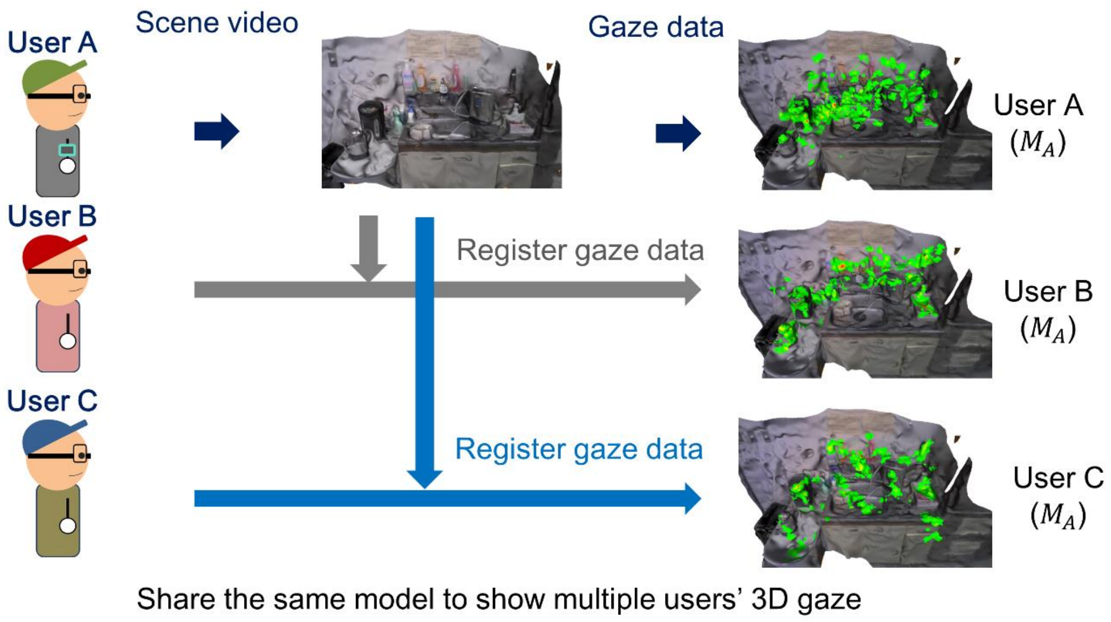


Fig. 3-17 Flowcharts of visualizing each user's gaze data on the model reconstructed from user's own recording: (a) text explanation and (b) a visualized plot

Take Fig. 3-17 for example. To realize the idea, the main challenge is to calculate the pose of camera centers of User B's and User C's video frames under the coordinate system of the model M_A. Once this is accomplished, the same process of calculating intersection lines and determining the intersected triangles on M_A is repeated. User B's and User C's 3D gaze fixation can then be shown on M_A.

Fig. 3-18 and Fig. 3-19 indicate the general procedure and the schematic plot of the calculation, respectively. Now, take only two users, User A and User B, for explanation, and assume they wear the same pair of eye tracking glasses and observe the same environment. User A's image set $\{I_i^A\}$ is chosen as the base.

After using the SfM method, densifying, and meshing, we obtain the base mesh model M_A for sharing and the pose of camera centers $\{\boldsymbol{c}_i^A\}$ corresponding to User A's video frames under the coordinate system of M_A, Σ_A.

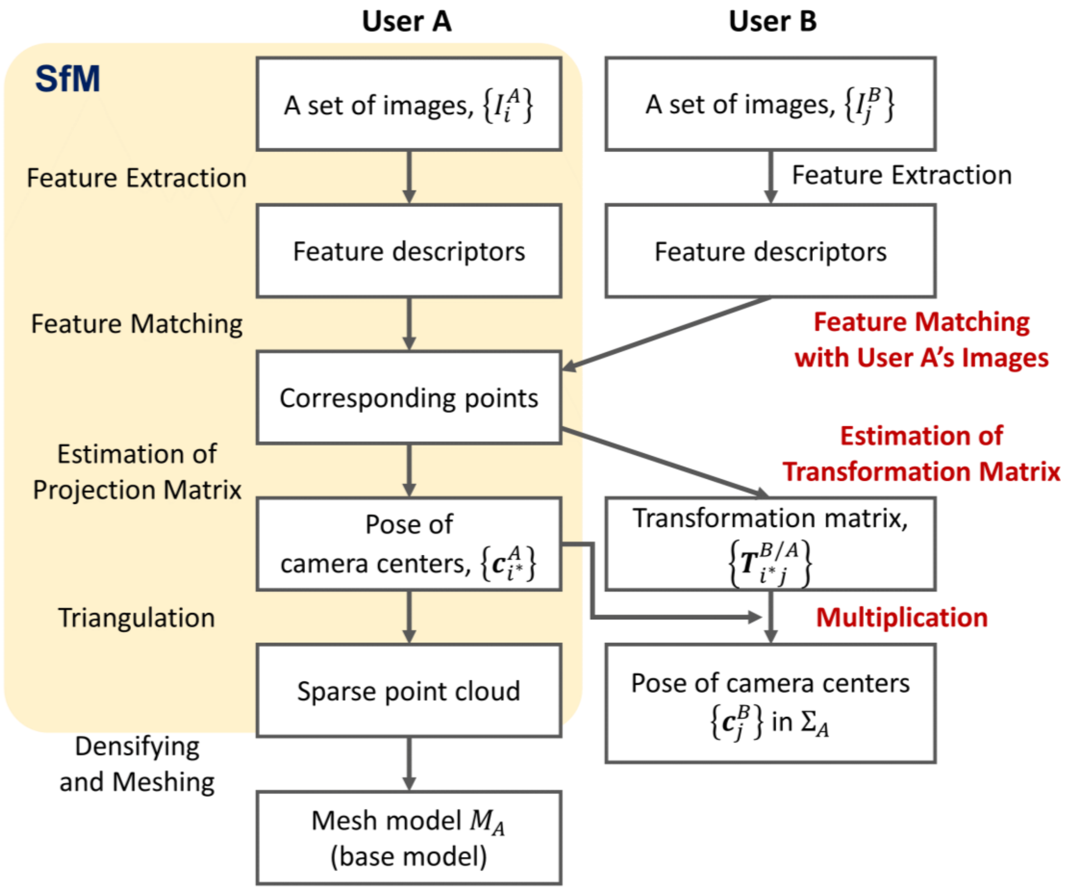

User A
User B
SfM
A set of images, $\{I_i^A\}$
A set of images, $\{I_j^B\}$
Feature Extraction
Feature Extraction
Feature descriptors
Feature descriptors
Feature Matching
Feature Matching with User A's Images
Corresponding points
Estimation of Projection Matrix
Estimation of Transformation Matrix
Pose of camera centers, $\{\boldsymbol{c}_{i^*}^A\}$
Transformation matrix, $\{\boldsymbol{T}_{i^*j}^{B/A}\}$
Triangulation
Multiplication
Sparse point cloud
Pose of camera centers $\{\boldsymbol{c}_j^B\}$ in Σ_A
Densifying and Meshing
Mesh model M_A (base model)

Fig. 3-18 Process of finding camera centers under the coordinate system of the base model M_A

Next, for each User B's video frame I_j^B, to estimate its pose of the camera center $\boldsymbol{c}_j^B$ in Σ_A, as shown in Fig. 3-19, the system estimate a transformation matrix $\boldsymbol{T}_{i^*j}^{B/A}$ that maps $\boldsymbol{c}_{i^*}^A$ of some appropriate $I_{i^*}^A$ to $\boldsymbol{c}_j^B$ based on the two-view geometry [74], as explained below. $\boldsymbol{c}_j^B$ in Σ_A could be derived by multiplying the transformation matrix $\boldsymbol{T}_{i^*j}^{B/A}$ and the known $\boldsymbol{c}_{i^*}^A$, as shown in Eq. 6. $I_{i^*}^A$ is chosen from $\{I_i^A\}$, which has the largest overlap (similarity) with I_j^B.

$$\boldsymbol{c}_j^B = \boldsymbol{T}_{i^*j}^{B/A} \times \boldsymbol{c}_{i^*}^A \tag{6}$$

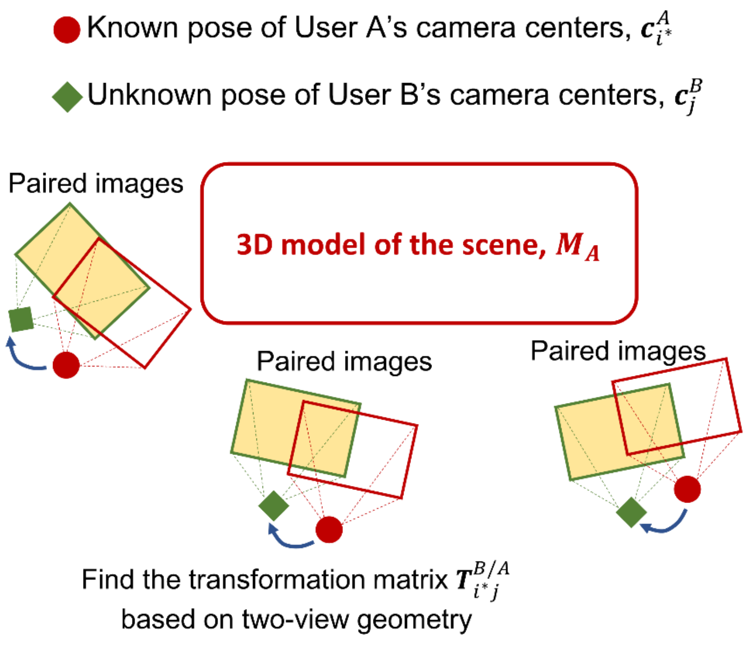

● Known pose of User A's camera centers, $\boldsymbol{c}_{i^*}^{A}$
◆ Unknown pose of User B's camera centers, $\boldsymbol{c}_{j}^{B}$
Paired images
3D model of the scene, M_A
Paired images
Paired images
Find the transformation matrix $\boldsymbol{T}_{i^*j}^{B/A}$
based on two-view geometry

Fig. 3-19 Schematic plot of determining camera centers under the coordinate system of the base model M_A

With respect to the two-view geometry, given the correspondence between two overlapping images of the same object acquired from different locations, the relative pose of the camera center (i.e., relative camera orientation and translation) could be derived. Here, the relative pose of camera centers acts as the transformation matrix $\boldsymbol{T}_{i^*j}^{B/A}$ that indicates the pose of User B's camera center relative to the pose of User A's camera center $\boldsymbol{c}_{i^*}^{A}$. Fig. 3-18 shows that to determine the $\boldsymbol{T}_{i^*j}^{B/A}$, feature extraction of two image sets, $\{I_i^A\}$ and $\{I_j^B\}$, and feature matching between them are first conducted to find some User A's image $I_{i^*}^{A}$ that corresponds to User B's image I_j^B. Because two users observe the same scene, for each User B's image, it is expected to find at least one User A's image that possesses a zone that is highly overlapped with the scene in User B's image. Once the corresponding image is found using the two-view geometry, the transformation matrix $\boldsymbol{T}_{i^*j}^{B/A}$ can be evaluated. Thus, the pose of User B's camera center $\boldsymbol{c}_j^B$ in Σ_A can be determined by multiplying the known pose of User A's camera center $\boldsymbol{c}_{i^*}^{A}$ and the transformation matrix $\boldsymbol{T}_{i*^*j}^{B/A}$ represented by homogeneous coordinates. This process is the same idea as registering new images $\{I_j^B\}$ into the existed model M_A, and we use the COLMAP software to address this image registration.

Then, using the above-mentioned method in Section 3.4.3, we can determine User B's 3D gaze fixation on the base model M_A. The connection of the pose of User B's camera center $\boldsymbol{c}_j^B$ and the 3D gaze point derived from Eq. 3 in Σ_A generates User B's line of sight in Σ_A to determine the intersection with M_A. Thus, we are able to produce User B's 3D gaze fixations in M_A. Furthermore, by applying the same concept to additional users who observe the same environment, the feature descriptors of their image sets can be matched to the feature descriptors of the image set of the base model M_A to determine the corresponding pose of the camera centers and to eventually visualized all users' 3D gaze fixation on the same model M_A. In addition, because all users' gaze data are shown on an identical model, only one 3D model of the scene is needed to be reconstructed instead of reconstructing multiple models from all users' recordings.

3.6 Experiments of Three-Dimensional Gaze Visualization System

We conducted some experiments to verify the proposed methodology and to discuss the functionality of our system. In this study, the specification of the PC used to generate 3D gaze visualization is show in Table 3-1. To speed up the process of 3D reconstruction, the COLMAP and OpenMVS software use the GPU to extract SIFT features of images and to refine the mesh model, respectively. Moreover, in our experiments, the number of frames for 3D scene reconstruction is restricted to approximately 200 because of the GPU computation in OpenMVS can handle approximately 200 images at most, which is related to the memory size of the GPU.

Table 3-1 Specification of the hardware for data processing

Processor	Intel® Core™ i7-7700 CPU @ 3.60GHz
RAM	64.0 GB
GPU	NVIDIA Geforce GTX 1080 Ti (Memory size: 11.0 GB)

3.6.1 System Verification

Two experiments were conducted to verify whether the evaluated positions of 3D gaze fixation are displayed on the target spots that a user paid attention to. The first experiment required the user to pay more attention to six fixed spots in the environment, as shown in Fig. 3-20(a). During the observation, the user moves from left to right and then from right to left, with the change of the target spot that the user paid attention to. The video is 53.6 sec long and contains 1,340 frame images. We took 224 frames ($n_s = 6$) for 3D scene reconstruction. In Fig. 3-20(b), it can be seen that most 3D gaze fixations are displayed on the specified six places. The average distance between target and measured positions is 6 mm, and the standard deviation is averagely 8 mm.

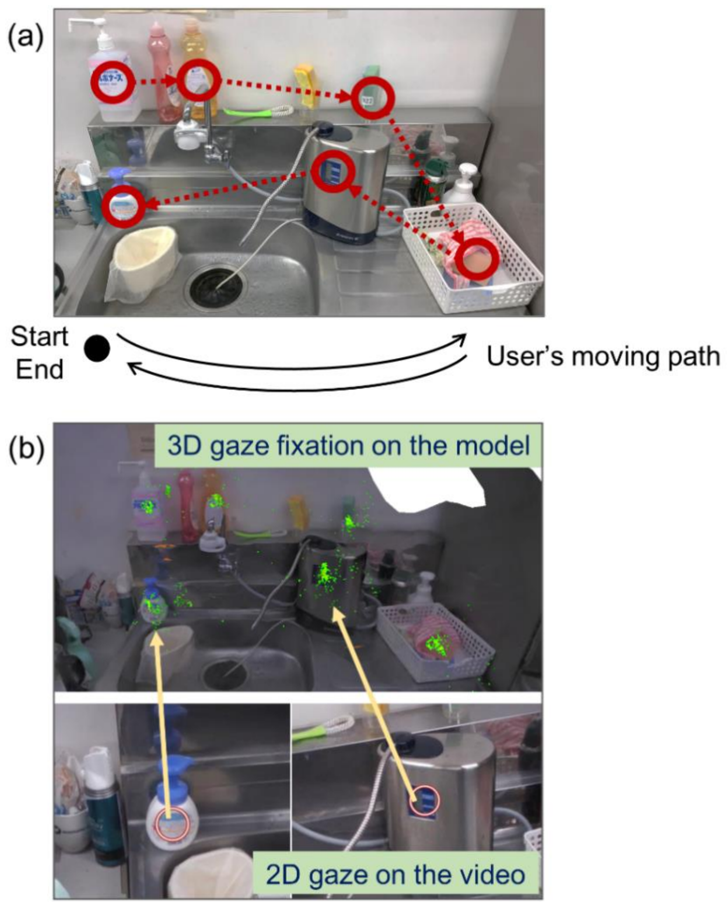


Fig. 3-20 System verification performed by observing six fixed spots: (a) target spots and the (b) evaluated result

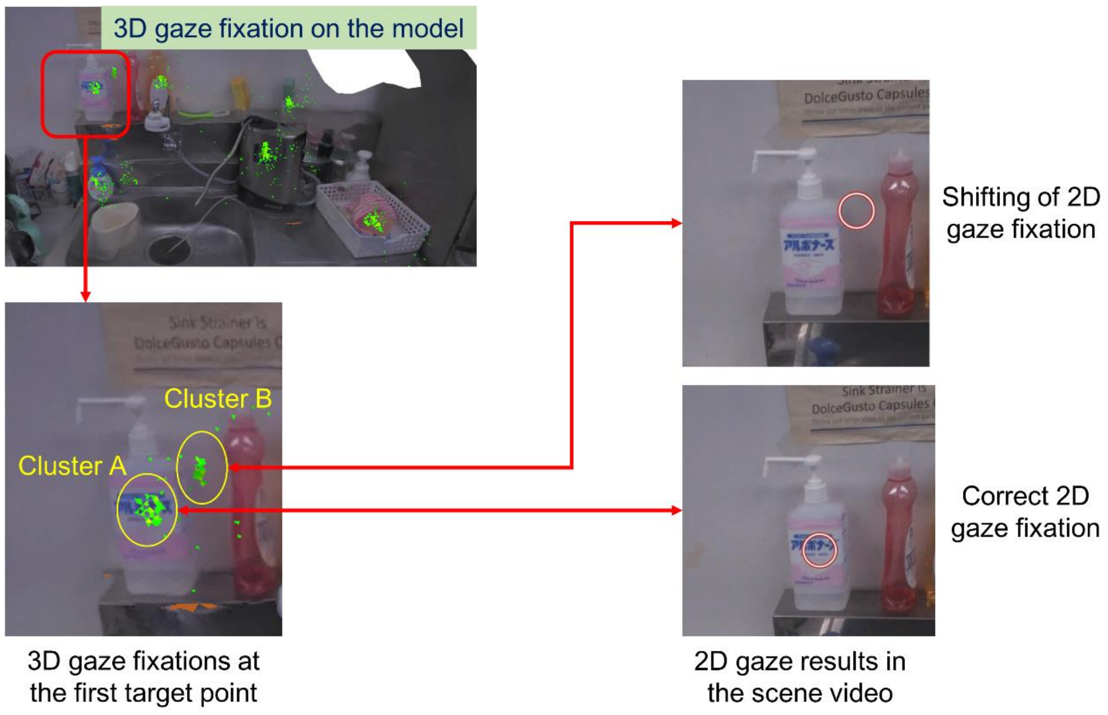

3D gaze fixation on the model
Cluster B
Cluster A
Shifting of 2D gaze fixation
Correct 2D gaze fixation
3D gaze fixations at the first target point
2D gaze results in the scene video

Fig. 3-21 Case of fixed spots: error of 3D gaze fixations from the shifting of 2D gaze

For the first target observation point as shown in Fig. 3-21, in the 3D gaze results, we see that there are two clusters A and B near the target point. 3D gaze fixations in Cluster A are the 3D gaze results corresponding to the correct 2D gaze fixations shown in the scene video. However, 3D gaze fixations in Cluster B are shifted approximately 5 cm away from the target point even when the user was definitely looking at the target point. This shifting results in the 3D model derive from the shifting results recorded in 2D gaze results that may be caused by the user's gaze angle larger than 15°. The high performance of Tobii eye tracking glasses is based on the gaze angle smaller than 15°. Gaze angle larger than 15° may cause larger errors in 2D gaze results that lead to larger errors in 3D. To avoid this problem, the user should move his/her head and body more frequently instead of large rotational movement of eyeballs.

The other experiment of system verification, shown in Fig. 3-22, required the user to observe along the edge of the sink back and forth. The video is 39.3 sec long and contains 982 frame images. 246 frames ($n_s = 4$) were taken into the 3D reconstruction process. The experimental result shows that 92% of evaluated 3D gaze fixation were located along the designated edges. Preliminarily, these two experiments show that most 3D gaze results using the proposed system are reliable.

However, Fig. 3-22(c) shows that as we observed the result of the edge case from another view, some estimated gaze positions were not on the edge of the sink but at its bottom. This derived from poor 3D reconstruction for that edge and gaze angle larger than 15°. The area marked with a dotted line is part of the sink edge where the corresponding meshes were poorly, or were not, reconstructed (the edge width shrinks by approximately 50%) because of its plain texture and insufficient images from multiple views for 3D reconstruction. This suggests that during the procedure of finding an intersected triangle, starting from the camera center, through user's line of sight, we were not able to find an intersection in the dotted area because there were no triangles; eventually, an intersection at the bottom of the sink was determined as we continued to move along the line of sight. Moreover, in Fig. 3-23, the other error source is the user's gaze angle larger than 15° as described in the case of observing fixed spots.

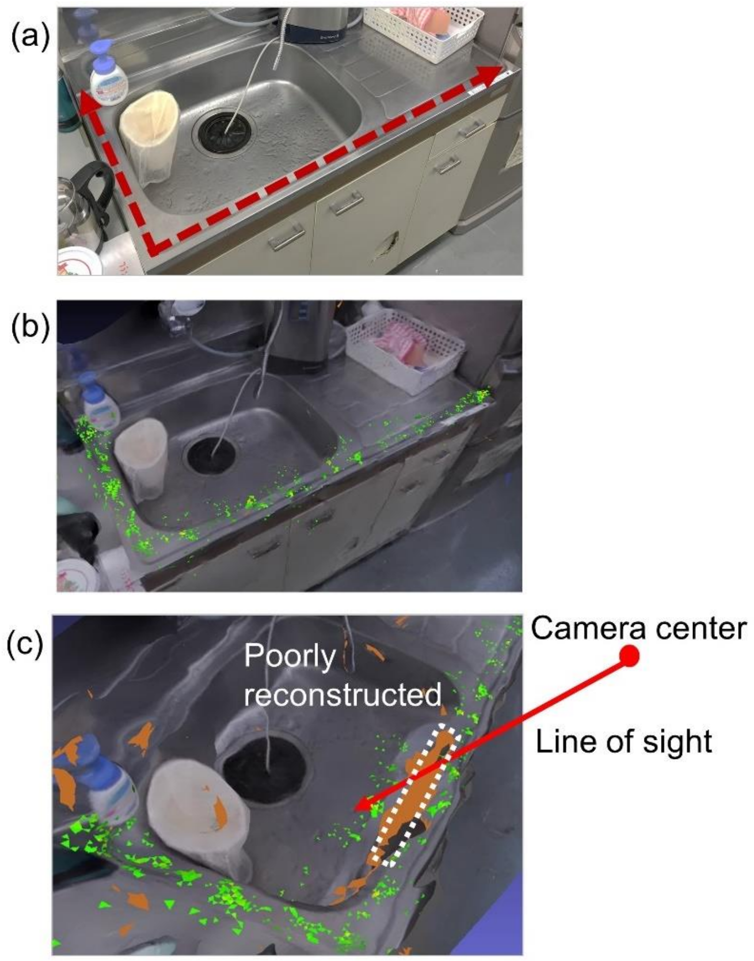


Fig. 3-22 System verification performed by observing the edge of the sink: (a) target edges, (b) evaluated result, and (c) different view of the evaluated result

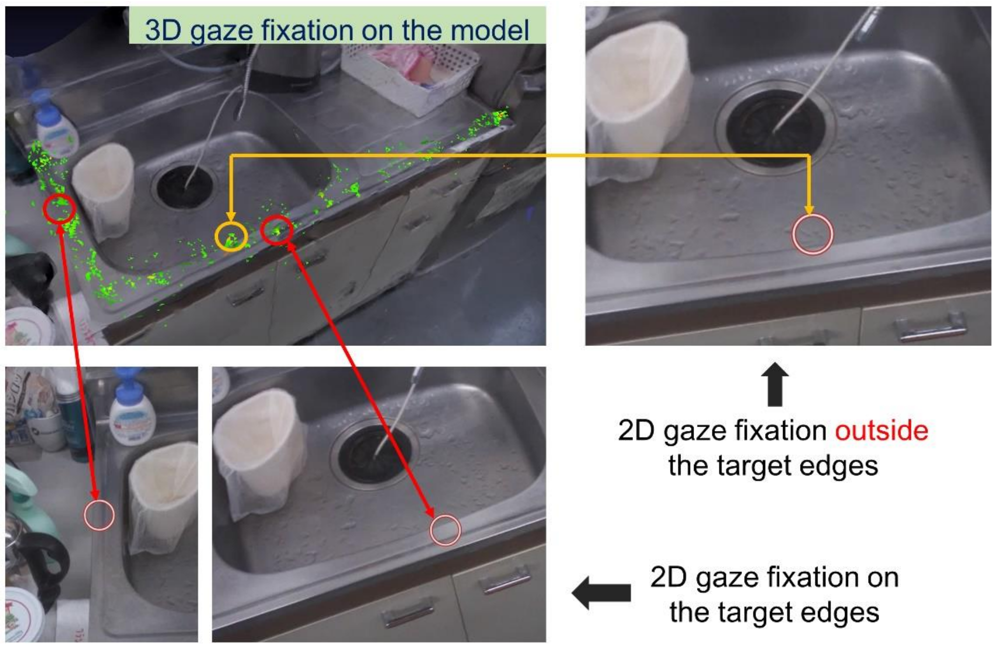

3D gaze fixation on the model
2D gaze fixation outside the target edges
2D gaze fixation on the target edges

Fig. 3-23 Case of edges: error of 3D gaze fixations from the shifting of 2D gaze

3.6.2 Gaze Difference between Multiple Users

To compare the gaze difference between multiple users, three users A, B, and C wore the same pair of Tobii eye tracking glasses and were asked to observe the same environment, i.e., the sink and surroundings, one by one. Fig. 3-24(a) shows the scene of the experimental environment. The users were restricted to observe the sink from the front and left side for the preliminary test. After the users finished the observation, the recorded video and the gaze data from the eye tracking glasses were input into the system for 3D reconstruction and to display 3D gaze fixation on the reconstructed model using 10 iterations of the application of the diffusion filter, as shown in Fig. 3-24(b), (c), and (d). Table 3-2 shows the information of each user's eye tracking recording, and the number of frames to generate the scene models M_A, M_B, and M_C from each user's recording. The processing time for each recording is shown in the upper part of Table 3-3. Thus, using this way for the three users observing the same scene, the system took totally 6 hours to demonstrate their 3D gaze results.

Table 3-2 Information of the scene videos

Video source	Video length [s]	Total frames	n_s	No. of frames for 3D reconstruction
User A	55.09	1379	6	230
User B	40.70	1019	5	204
User C	48.41	1212	6	202

Table 3-3 Processing time of each stage (minute)

Video Source (Model)	3D scene reconstruction		Image registration[a]	3D gaze determination	Total
	Sparse point cloud generation (COLMAP)	Mesh model generation (OpenMVS)			
	Users' gaze on their respective model				
User A (M_A)	2.85	118.47	5.95	8.18	135.45
User B (M_B)	2.4	94.40	4.88	7.1	108.78
User C (M_C)	2.65	94.33	5.68	8.52	111.18
	Users' gaze on the same model, M_A				
User B (M_A)	-	-	9.27	7.90	17.17
User C (M_A)	-	-	9.83	8.4	18.23

[a]Register the images that are not used for 3D reconstruction into the model to obtain corresponding camera poses and to determine 3D gaze fixation

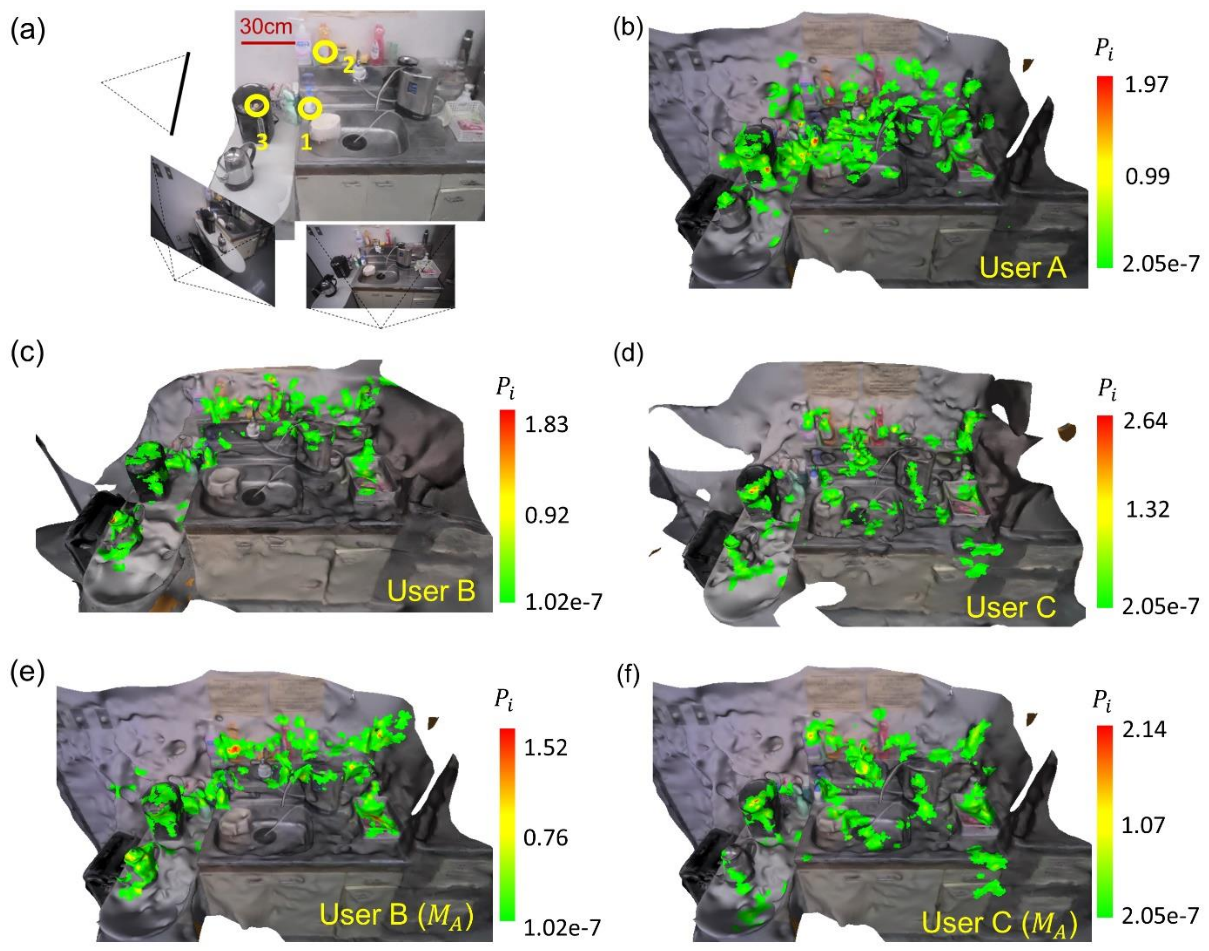


Fig. 3-24 Visualization of 3D gaze fixation for three users: (a) experimental environment, (b)(c)(d) three users' 3D gaze visualization on the 3D models reconstructed from their own recorded scene video, (e)(f) display of User B's and C's 3D gaze fixation on M_A

Next, to compare the gaze distribution, we used User A's reconstructed model M_A as the base model to visualize the other two users' gaze data on M_A, as shown in Fig. 3-24(e) and (f). The processing time is shown in the lower part of Table 3-3. Using this method, the system first generated M_A with 135.45 minutes and then registered User B′s and C′s gaze data to M_A with 35.4 minutes. Thus, it took 2.8 hours to generate all the results, which saved 50% of the processing time.

Table 3-4 Fixation count P_i of the three spots

	Spot		
Gaze data	1	2	3
User B [Fig. 3-24(c)]	0.072	1.149	0
User B (M_A) [Fig. 3-24(e)]	0.476	1.447	0
User C [Fig. 3-24(d)]	0	0	1.955
User C (M_A) [Fig. 3-24(f)]	0	0.003	2.074

Table 3-4 shows the quantitative performance of the registration. We chose three spots marked in Fig. 3-24(a) and, for the same user, compared the fixation count P_i between the 3D model reconstructed from their own scene video and M_A. Spots 1, 2, and 3 were the places where Users A, B, and C paid more attention, respectively. Table 1 indicates that, for the same user, there was a similar distribution of the fixation count P_i on the different reconstructed 3D models. By displaying User B's and C's gaze data on M_A, we could determine the difference between the same gaze information on the different models. This occurred owing to the different quality of the models. Different recorded videos generated models with different qualities. Although the videos from different users recorded the same environment, the frame images were taken from different views of the environment. In addition, the existence of textureless objects (e.g., the wall and the table) resulted in the deformation of objects and creation of uneven surfaces owing to few detected feature points on the textureless objects. These issues changed the position of intersected points on the models. Nevertheless, the parts to which a user paid more attention were still marked as the key parts on the model M_A.

3.6.3 3D Gaze Visualization in a Room-Scale Environment

Fig. 3-25 shows a room-scale observation. A user walked around in a room and paid attention to decorations near three inside walls of the room. The video is 126.8 sec long and contains 3,170 frame images. 212 frames were taken for 3D scene reconstruction. Fig. 3-25(b) shows a panorama of the room, and the dotted lines indicate the junction of two walls. A 3D model of three inside walls in the room was reconstructed with 3D gaze fixation, and 10 iterations of the diffusion filter with $\lambda = 0.8$ were applied. To mitigate the accumulation of error, which is common in image-based 3D reconstruction, the COLMAP software applies bundle adjustment to minimize reprojection errors. In this room-scale experiment, the reprojection error is averagely 1 pixel. Fig. 3-25(a) demonstrates the potential of this system to be applied to larger and more complex environments.

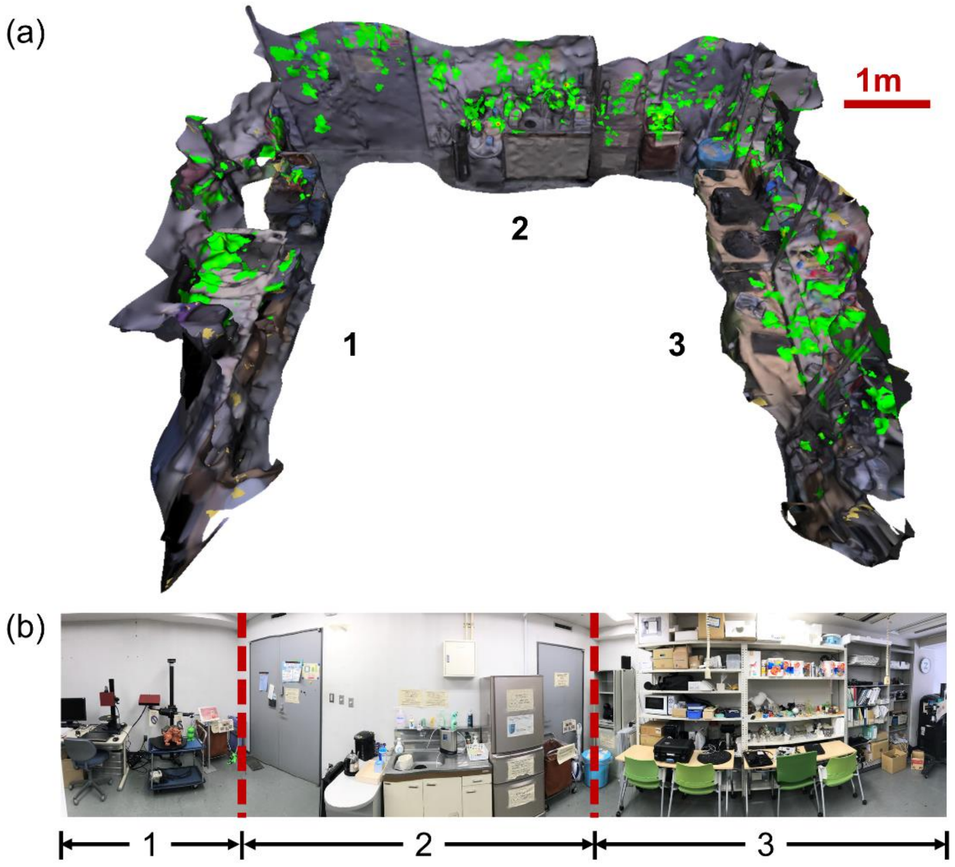


Fig. 3-25 3D gaze visualization in a room-scale model: (a) model of the room with 3D gaze fixations and a (b) panorama of the room

However, in a room-scale environment, it is also possible to perform gaze analysis with 2D images. Fig. 3-26 shows the heatmap results for the panorama of the room, which was generated by Tobii Pro Lab (i.e., a commercial eye tracking software for analyzing and visualizing 2D gaze data from Tobii eye tracking glasses). The panoramic image was acquired by an extra RGB camera and input into the Tobii Pro Lab software. By comparing each frame image in the scene video and recorded gaze data, the software evaluated the corresponding gaze fixations and visualized them on the panorama using the heatmap. Although it appears that 2D gaze visualization is an adequate solution, it is still not reliable even in this case. Fig. 3-27(a) is a frame image with gaze fixation acquired from the scene video, and its corresponding gaze fixation in the panorama, determined by the Tobii pro lab, is shown in Fig. 3-27(c). Fig. 3-27(d) shows the heatmap result in the panorama. According to the frame image of the scene video in Fig. 3-27(a), a user was looking at the inside of a brown bin without a cover, but after the analysis of the Tobii pro lab software, the same gaze fixation is displayed on the outside of the bin. This error is inevitable because the inside image of the bin does not exist in the panorama. This shows that if we want to obtain such gaze fixation results on 2D images, we need to have multiple images with various views of the environment. However, as the scale of the environment becomes larger, the number of required images will exponentially increase, which will make the analytic work more difficult. Fig. 3-27(c) shows the result of 3D gaze visualization that demonstrates that the user paid attention to the inside rather than the outside.

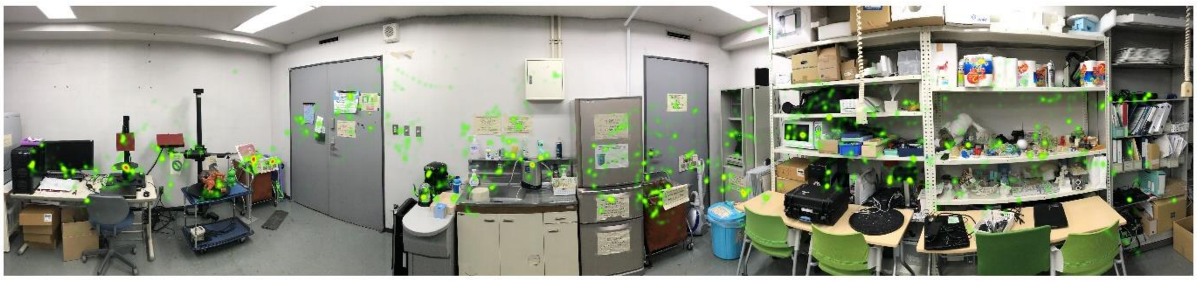


Fig. 3-26 2D gaze visualization on the panorama of the room

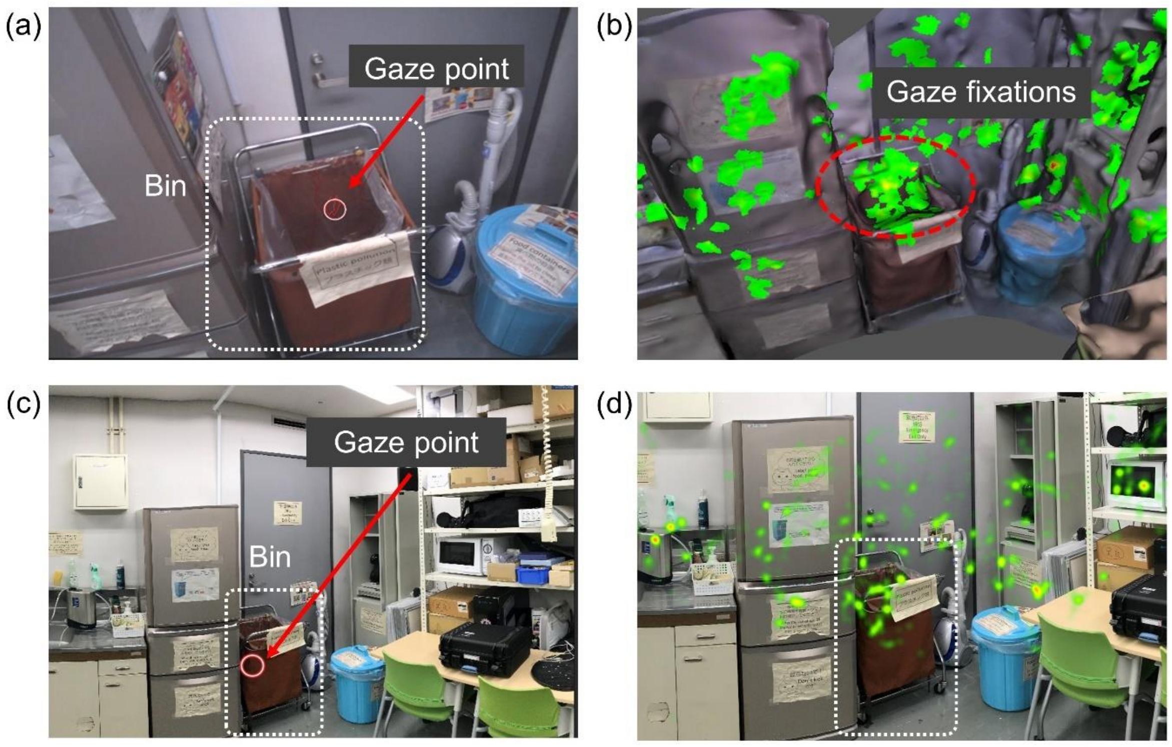


Fig. 3-27 Comparison of gaze estimation between (a) the frame image from the scene video, (b) proposed 3D gaze visualization, and (c) the panorama. (d) is the heatmap form of 2D gaze in the panorama.

The room-scale experiment shows an advantage of 3D gaze visualization over 2D gaze information is its spatial extensibility. Using the 3D reconstruction from the scene video, a larger model (e.g., a room or even a building) can be generated to easily display and analyze gaze fixations. However, it is difficult to achieve the same effects with 2D images.

3.7 Summary

In this study, we propose a system that uses the eye tracking glasses to demonstrate user's 3D gaze fixation on a 3D model of the scene. Though other existing systems similar to it employ extra sensors, our system employs only the eye tracking glasses without any other sensors. The eye tracking glasses record all the necessary information for 3D gaze visualization: the scene video and user's 2D gaze data. Through image-based 3D reconstruction, the scene video can be used to reconstruct the 3D environmental model by using the COLMAP and OpenMVS software. To reduce the processing time, we take a part of frame images for 3D scene reconstruction by downsampling the video. Next, by estimating the pose of camera centers and gaze data, we can determine the 3D gaze fixations on the reconstructed model of the scene by drawing the intersection line, which is a user's line of sight. The preliminary results of the experiments indicate that user's 3D gaze fixation on the reconstructed model can be displayed at the targets with the accuracy of 6 mm and the standard deviation of 8 mm.

Moreover, to analyze gaze differences between multiple users, we use the image registration method to find all users' 3D gaze fixations on the same model of the scene. In the experiments of the three users’ 3D gaze visualization, 50% of the processing time can be saved. By sharing the same model, we can more efficiently visualize multiple users’ 3D gaze for worker’s assessment and training. In addition, a room-scale experiment was conducted. It shows that the 3D gaze visualization can demonstrate the structure of the environment and may give more reliable results than 2D gaze visualization.

Since only a pair of eye tracking glasses is used and all the necessary data are collected during eye tracking experiments, we can save time and human resources without scanning the environment again by other devices. This methodology exhibits a considerable potential for the applications related to the evaluation of user's attention in large and complex environments such as for the instruction of construction inspectors or for the aesthetic evaluation of interior decoration.

Chapter 4
Conclusion and Future Work

4.1 Summarization of the work

With the development of smart glasses, it can be equipped with various sensors and high-performance microprocessors. This kind of wearable device has become a promising tool for field service assistance. In this research, we focus on the AR-based indoor field service assistance system which may be realized by using AR glasses with eye tracking sensors. However, since the development of such smart glasses is still in its infancy, we use AR glasses and eye tracking glasses, which are available in the market, and develop two prototype systems for the assistance of indoor field service applications that the worker needs to move in room-scale environments. AR glasses are responsible for the assistance during assembling operations, and eye tracking glasses can be used for worker′s skill assessment and training. The main achievement in our work can be summarized as follows:

- ✓ For AR glasses, we develop an AR assembly assistance system with misassembly detection in real time.

- ✓ For eye tracking glasses, we visualize multiple users′ 3D gaze fixations effectively for gaze comparison.

The summarization and achievement of each study is as follows:

In the part of AR glasses (Chapter 2)
We propose an AR assembly assistance system with misassembling detection in real time. By using the sensors and microprocessors in the AR glasses, a standalone AR system has been developed for assembly assistance. We study the issues of coordinate calibration and efficient misassembly evaluation to make virtual parts displayed at desired installing locations and to evaluate assembly errors in real time, respectively. The developed system is experimentally validated and demonstrated in desktop applications and the room-scale environment.

To realize such AR system, we integrate the existing functions in the AR glasses and our proposed methods. For example, coordinate calibration is composed of rough alignment (user's hand manipulation) and precise alignment (the point-to-plane ICP algorithm). The process of rough alignment can be fast, but the position error may be a few centimeters. Moreover, precise alignment can achieve millimeter-level accuracy, but the initial position and orientation of two point clouds should be similar to prevent from local convergence; that is, initial alignment is needed and should not be poor. Thus, we propose an idea of the integration of the two alignment methods to achieve coordinate calibration of millimeter-level accuracy and simultaneously make the whole process done within a few seconds.

Furthermore, to evaluate assembly errors in real time, we make efforts to reduce the computation time by comparing the HoloLens-generated depth maps (in 2D space) instead of in 3D space. By comparing the relationship between the pixel values of the depth maps, in our preliminary implementation, the evaluation of misassembly is performed at 30 fps, which is sufficiently fast to warn the worker once misassembly occurs. The evaluation result is visualized by coloring virtual parts. Moreover, the system can detect the position error of assembly within ±1 cm, which can be expected to be used for room-scale environments.

In the part of eye tracking glasses (Chapter 3)
We propose a 3D gaze visualization method by converting the recorded gaze data in the scene video (2D space) into 3D space. Without using external scanners to perform another 3D scanning of the environments, the necessary image data for 3D reconstruction can be recorded by the eye tracking glasses when the worker is working. The proposed system is experimentally validated and demonstrated for a room-scale environment. User's focus of attention can be marked in a 3D model of the scene in the form of heatmaps, which are commonly used to display the user's visual attention.

To realize such 3D gaze visualization more effectively and efficiently, instead of directly inputting all frame images in the recorded scene video into the existing software for 3D scene reconstruction, we divide the fame images into two groups in advance. One is used for 3D scene reconstruction, and the other is used to find the camera centers to determine 3D gaze fixations. With the idea of downsampling the number of frame images for 3D scene reconstruction, the reconstruction time can effectively decrease from dozens of hours to a few hours. Moreover, since the scene video and gaze data are collected by different devices at different frequency, we estimate the corresponding gaze information for each frame image by linearly interpolating the gaze data based on the time stamps of the gaze data and of the frame images.

Furthermore, to compare multiple users′ 3D gaze results more effectively, we further propose an approach to visualize all users′ gaze in the same model of the scene. In our preliminary experiments of three users, by sharing the same 3D model of the scene, the total generation time of the three users′ 3D gaze results can decrease by 50%. In comparison to the typical 2D eye tracking applications that workers are required to observe a static image of a working field, our system allows the workers to move in the field and work as usual. It is expected to compare practical working performance between experienced and novice workers such as inspecting a construction field.

In summary, we use existing AR glasses and eye tracking glasses to develop the systems for field service assistance. The AR assembly assistance system gives assembly indications and evaluates misalignment when the worker is performing an assembly operation. 3D gaze visualization is possible to compare multiple workers′ practical performance at work. This is beneficial to train novice workers and skill assessment. In the future, these two systems can be integrated in a single AR glasses with eye tracking sensors and form an AR-based field service assistance system. We can expect that using a pair of such smart glasses can assist in field service applications and training.

4.2 Future Perspective

This work mainly includes the development of the AR assembly assistance system and 3D visualization of users′ gaze fixations on objects. These systems are experimentally validated and are possible to be used in indoor field service assistance of room-scale environments. In the future, we will continue studying and improving those research topics and make efforts to integrate the two systems into the same pair of smart glasses. The following list our future work in three parts:

- **In the part of AR assembly assistance (AR glasses)**

  (a) <u>Develop more robust indicator for evaluation of misalignment.</u>

  In Sec. 2.5, to evaluate misalignment between real and the virtual objects, we focus on the comparison of the depth values between the depth maps of the real and virtual objects. We use centered cosine similarity as our indicator to evaluate the misalignment, and in our preliminary experiments, the accuracy of the misalignment error can be within ±1 cm. As validated, the centered cosine similarity method is invariant to the shift and scaling of data set. However, a part of the real object may be poorly scanned because of multipath interference in the time-of-flight depth camera or scanning from a shallow angle. It will influence the evaluation results of misalignment.

  Hence, it is necessary to develop a more robust indicator to the poorly scanned point cloud. In the future, we can try to integrate with other methods, such as edge detection in the depth map, comparing silhouette of objects, and object tracking. Each method can give an evaluation score. By fusing various methods, we can then determine an evaluation function as a new indicator for misassembly evaluation. Simultaneously, we still have to take the processing time into account to confirm the evaluation can be performed in real time.

  Another issue of misalignment evaluation is occlusion problems with hands and other objects. In our development, the occlusion problems are not considered. When evaluating misalignment for each part of an assembly, there should not exist any other objects between the HoloLens and the target part of the assembly. This will increase the time of assembling operations. To solve the occlusion problems of user′s hands and of other parts of the assembly, it is possible to remove scanned hands and parts from the depth maps by hand tracking and object tracking. Then, we can take remaining pixels in the depth maps to evaluate whether there is occurrence of misalignment.

In addition, the scanned point cloud of nearby objects such as table surfaces also influences the misalignment evaluation and make the indicator′s values asymmetric when the misalignment occurs in different directions. To solve this problem, we may perform object detection in depth maps and take only pixels that belongs to the target object to do evaluation. This will mitigate the influence given by nearby objects′ point cloud.

(b) Improve the design of the graphical user interface (GUI) and AR assembly instructions.

In our implementation of the AR assembly assistance system, we mainly focus on the development of the functions, and the GUI is simply designed with a few aligned buttons to execute the functions, such as performing point-to-plane ICP for coordinate calibration and showing the next virtual part for assembly. To assist workers in assembly operations using AR technology more smoothly, clear, concise, effective user interfaces can help the workers understand how to read and use the AR assembly assistance system. Thus, we will make efforts to improve our GUI design.

Moreover, we would like to design AR assembly instructions to help the user find the virtual parts. For our room-scale demonstration in Sec. 2.6.4, decorations are arranged in the different locations of an 8 m^2 room. When the system sequentially displays each decoration in the room, the user can have no difficulty in finding the target virtual object. However, given that the user is in a much larger environment, the user may not be able to smoothly find where the target virtual object (i.e., the next installing location) is. In that case, it will be beneficial if the system can give instructions to help the user find the location. Since the AR glasses of HoloLens can position the user′s position in the environment, we can design an AR arrow to indicate the direction of the installing location according to the position of the target virtual object relative to the user′s position.

■ **In the part of 3D gaze visualization (eye tracking glasses)**

(a) Reduce the time required for 3D scene reconstruction.

In Table 3-3 in Sec. 3.6.2, to generate User A′s 3D gaze visualization, we perform a series of processing stages. In particular, the processing time for 3D scene reconstruction accounts for approximately 87% of the entire processing time. Thus, there is still room for improvement to accelerate the process of gaze analysis. Further downsampling the number of frame images in chronological order is one way to reduce the number of images used for 3D reconstruction. However, if a user walks around in an environment and frequently observes the same object at different times, frame images that have the same view will still be selected out. Those frame images can not provide new information of the scene for 3D reconstruction. To avoid such redundancy, a possible way is to remove the frame images that have the same or similar view from the process of 3D reconstruction. Thus, the time for 3D reconstruction will decrease.

(b) Improve the completeness of the 3D model of the scene for multiple users′ 3D gaze visualization.

In our preliminary experiment in Sec. 3.6.2, multiple users′ (Users A, B, and C) gaze fixations are visualized by the same 3D model of the scene which is constructed from User A′s scene video. With this approach, if now Users B and C observe some objects that are not observed by User A, it can be expected that those objects will not be reconstructed in the model of the scene, and the generated 3D gaze results will not be reliable. To solve this problem, we can take all users′ scene videos into account and reconstruct a 3D model of the scene that covers all users′ focus of attention as the main model for 3D gaze visualization.

■ **Integration of AR assembly assistance and 3D gaze visualization systems**

(a) Integrate the developed two systems into a pair of smart glasses.

In the future, we will deploy the two subsystems into a pair of AR glasses with eye tracking sensors, develop an AR-based field service assistance system, and verify the performance of the integrated system in AR-based field service applications. Microsoft HoloLens 2 (the 2nd generation of HoloLens) is possible commercial smart glasses for our purpose. HoloLens 2 is a pair of AR glasses with eye tracking sensors while HoloLens that we use does not have embedded eye tracking sensors.

After integrating the two systems, it is an issue to consider the interaction between eye tracking and AR technologies. For example, with user′s gaze information, it can be a controller to interact with virtual objects in AR environments. However, when the user is gazing some virtual object, it can be a challenge for application developers to identify whether the user just looks at it or wants to manipulate it by gaze.

Conversely, under AR environments, the user can see not only physical objects but also computer-generated virtual objects. It means that when we want to analyze the user′s gaze data, we should also take the user′s gaze fixations on virtual objects into account. Thus, for 3D gaze visualization, the 3D model of the scene will contain real objects, generated by image-based 3D reconstruction, and virtual objects, which can be directly input into the model of the scene if we can evaluate their positions in the model. Then a potential study for the AR glasses with eye tracking sensors is that how the AR assistance system influences the worker′s performance, and the user′s gaze information may be a method for analysis.

Reference

[1] H. L. Chi, S. C. Kang, and X. Wang, "Research trends and opportunities of augmented reality applications in architecture, engineering, and construction," *Automation in Construction*, vol. 33, pp. 116-122, August 2013.

[2] S. Webel, U. Bockholt, T. Engelke, N. Gavish, M. Olbrich, and C. Preusche, "An augmented reality training platform for assembly and maintenance skills," *Robotics and Autonomous Systems*, vol. 61, no. 4, pp. 398-403, April 2013.

[3] Fieldbit, https://www.fieldbit.net/

[4] A. Tang, C. Owen, F. Biocca, and W. Mou, "Comparative effectiveness of augmented reality in object assembly," In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI'03, pp. 73-80, New York, USA, 2003.

[5] O. Oda, C. Elvezio, M. Sukan, S. Feiner, and B. Tversky, "Virtual replicas for remote assistance in virtual and augmented reality," In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, UIST'15, pp. 405-415, November 2015.

[6] B. Schwald and B. de Laval, "An augmented reality system for training and assistance to maintenance in the industrial context," *Journal of WSCG*, vol. 11, no. 1-3, 2003.

[7] M. F. Alam, S. Katsikas, O. Beltramello, and S. Hadjiefthymiades, "Augmented and virtual reality based monitoring and safety system: A prototype IoT platform," *Journal of Network and Computer Applications*, vol. 89, pp. 109-119, 2017.

[8] Trimble Mixed Reality, https://mixedreality.trimble.com/

[9] S. Hasanzadeh, B. Esmaeili, and M. D. Dodd, "Measuring construction workers' real-time situation awareness using mobile eye-tracking," *Construction Research Congress 2016*, pp. 2894-2904, 2016.

[10] T. Tien, P. H. Pucher, M. H. Sodergren, K. Sriskandarajah, G.-Z. Yang, A. Darzi, "Eye tracking for skills assessment and training: a systematic review," *Journal of Surgical Research*, vol. 191, no. 1, pp. 169-178, September 2014.

[11] S. Hasanzadeh, B. Esmaeili, and M. D. Dodd, "Measuring the impacts of

safety knowledge on construction workers′ attentional allocation and hazard detection using remote eye-tracking technology,″ *Journal of Management in Engineering*, vol. 33, no. 5, pp. 04017024, 2017.

[12] Tobii AR, https://ar.tobii.com/

[13] H.-J. Joo and H.-Y. Jeong, ″A study on eye-tracking-based Interface for VR/AR education platform,″ *Multimedia Tools and Applications* 79, pp. 16719-16730, 2020.

[14] R. Radkowski and S. Kanunganti, ″Augmented Reality System Calibration for Assembly Support with the Microsoft HoloLens,″ *In ASME 2018 International Manufacturing Science and Engineering Conference (MSEC 2018)*, College Station 2018.

[15] R. Azuma, ″A survey of augmented reality,″ Presence: *Teleoperators and Virtual Environments*, vol. 6, no. 4, pp. 355-385, 1997.

[16] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre, ″Recent advances in augmented reality,″ *IEEE Computer Graphics and Applications*, vol. 21, no. 6, pp. 34-47, 2001.

[17] Z. Lv, A. Halawani, S. Feng, S. U. Réhman, and H. Li, ″Touch-less interactive augmented reality game on vision-based wearable device,″ *Personal Ubiquitous Computing*, vol. 19, no. 3-4, pp. 551-567, July 2015.

[18] E. Z. Barsom, M. Graafland, and M. P. Schijven, ″Systematic review on the effectiveness of augmented reality applications in medical training,″ *Surgical Endoscopy*, vol. 30, no. 10, pp. 4174-4183, 2016.

[19] F. M. Dinis, A. S. Guimarães, B. R. Carvalho, and J. P. P. Martins, ″Virtual and augmented reality game-based applications to civil engineering education,″ *2017 IEEE Global Engineering Education Conference (EDUCON)*, pp. 1683-1688, Athens, Greece, April 2017.

[20] SIEMENS: Ease the delivery of complex assembly procedures for highly configurable products, https://www.plm.automation.siemens.com/global/ja/products/manufacturing-planning/augmented-reality-assisted-work-instructions.html

[21] M. D. Mura, G. Dini, F. Failli, ″An Integrated Environment Based on Augmented Reality and Sensing Device for Manual Assembly Workstations,″ *Procedia CIRP*, vol. 41, pp. 340-345, 2016.

[22] J. Alves, B. Marques, M. Oliveira, T. Araújo, P. Dias, B. S. Santos,

Comparing Spatial and Mobile Augmented Reality for Guiding Assembling Procedures with Task Validation," *In 2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pp. 1-6, Porto 2019.

[23] J. Sääski, T. Salonen, M. Hakkarainen, S. Siltanen, C. Woodward, and J. Lempiäinen, "Integration of design and assembly using augmented reality," *International Federation for Information Processing on Micro-Assembly Technologies and Applications*, vol. 260, pp. 395-404, 2008.

[24] G. Evans, J. Miller, M. I. Pena, A. MacAllister, E. H. Winer, "Evaluating the Microsoft HoloLens through an Augmented Reality Assembly Application," *In Proceedings of SPIE*, 10197, pp. 101970V, 2017.

[25] P. Hübner, K. Clintworth, Q. Liu, M. Weinmann, S. Wursthorn, "Evaluation of HoloLens Tracking and Depth Sensing for Indoor Mapping Applications," *Sensors*. 2020; 20(4):1021.

[26] HoloLens device specifications, https://www.beachtle.com/shop/medias/5a1430f79ce96955066d0b10.pdf?context=bWFzdGVyfHJvb3R8NjA0ODN8YXBwbGljYXRpb24vcGRmfGhlNC9oM2UvOTQ3MDY4ODYyNDY3MC5wZGZ8MmUwZjQ0YWMyYmUxMTk3MGE0MmJkNWQxNjI4NmU1MzFiN2Q2MDU4MDYxYjM0MmJiYjkzNzQ5NmY0Njc0YjJhZA

[27] Depth camera used in the HoloLens, https://www.microsoft.com/en-us/research/blog/microsoft-hololens-facilitates-computer-vision-research-by-providing-access-to-raw-image-sensor-streams-with-research-mode/

[28] Mixed Reality Toolkit, https://github.com/microsoft/MixedRealityToolkit-Unity

[29] Introduction to the HoloLens: Spatial Mapping, https://docs.microsoft.com/en-us/archive/msdn-magazine/2017/january/hololens-introduction-to-the-hololens-part-2-spatial-mapping#working-with-the-spatial-mesh

[30] API of MapImagePointToCameraUnitPlane() related to camera intrinsics of the HoloLens depth camera, https://github.com/microsoft/HoloLensForCV/blob/master/Shared/HoloLensForCV/CameraIntrinsics.cpp

[31] HoloLensForCV repository, https://github.com/microsoft/HoloLensForCV

[32] S. Foix, G. Alenyà, and C. Torras, "Lock-in Time-of-Flight (ToF) Cameras:

A Survey, " *IEEE Sensors Journal*, vol. 11, no. 9, pp. 1917-1926, 2011.

[33] A. Bhandari, M. Feigin, S. Izadi, C. Rhemann, M. Schmidt, and R. Raskar, "Resolving multipath interference in Kinect: An inverse problem approach," *SENSORS*, 2014 IEEE, Valencia, pp. 614-617, 2014.

[34] J. Mure-Dubois and H. Hϋgli, "Real-time scattering compensation for time-of-flight camera, " In *Proceedings of the ICVS Workshop on Camera Calibration Methods for Computer Vision Systems (CCMVS2007)*, 2007.

[35] T. Huang, K. Qian, and Y. Li, "All Pixels Calibration for ToF Camera," In *IOP Conference Series: Earth and Environmental Science*, vol. 170, no. 2, 2018.

[36]D. Sjöholm, "Calibration using a general homogeneous depth camera model," Dissertation, 2017.

[37] Y. He, B. Liang, Y. Zou, J. He, and J. Yang, "Depth Errors Analysis and Correction for Time-of-Flight (ToF) Cameras. Sensors," *Sensors*, vol. 17, 2017.

[38] Y. Chen and G. Medioni, "Object Modeling by Registration of Multiple Range Images," In *1991 IEEE International Conference on Robotics and Automation*, Sacramento, 1991.

[39] K. L. Low, "Linear least-squares optimization for point-to-plane ICP surface registration," Technical report, University of North Carolina at Chapel Hill, 2004.

[40] k-d tree, https://en.wikipedia.org/wiki/K-d_tree

[41] P. J. Besl and N. D. McKay, "A Method for Registration of 3-D Shapes," IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), vol. 14, no. 2, pp. 239-256, 1992.

[42] S. Rusinkiewicz and M. Levoy, "Efficient Variants of the ICP Algorithm," Proceedings of the International Conference on 3-D Digital Imaging and Modeling (3DIM), pp. 145-152, 2001.

[43] Point Cloud Library, https://pointclouds.org/

[44] Unity, https://unity.com/

[45] Generate depth maps in Unity, https://docs.unity3d.com/Manual/SL-CameraDepthTexture.html

[46] A. T. Duchowski, "Eye Tracking Methodology: Theory and Practice," Springer-Verlag, 2003.

[47] H. L. Kundel, C. F. Nodine, E. F. Conant, and S. P. Weinstein, "Holistic component of image perception in mammogram interpretation: Gaze-tracking study," *Radiology*, vol. 242, pp. 396-402, 2007.

[48] H. Song, J. Lee, T. J. Kim, K. H. Lee, B. Kim, and J. Seo, "Gazedx: Interactive visual analytics framework for comparative gaze analysis with volumetric medical images," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, pp. 311-320, 2017.

[49] H. Song, J. Yun, B. Kim, and J. Seo, "Gazevis: Interactive 3d gaze visualization for contiguous cross-sectional medical images," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, pp. 726-739, 2014.

[50] K. Chaika, M. Hayhoe, B. Sullivan, J. Pelz, N. Mennie, and J. Droll, "Predictive eye movements in squash," *Journal of Vision*, vol. 6, pp. 481-481, 2006.

[51] D. T. Mann, A. M. Wiliams, P. Ward, and C. M. Janelle, "Perceptual-cognitive expertise in sport: A meta-analysis," *Journal of Sport and Exercise Psychology*, vol. 29, pp. 457-478, 2007.

[52] M. L. Mele and S. Federici, "Gaze and eye-tracking solutions for psychological research," *Cognitive Processing*, vol. 13, pp. 261-265, 2012.

[53] M. Vidal, J. Turner, A. Bulling, and H. Gellersen, "Wearable eye tracking for mental health monitoring," *Computer Communications*, vol. 35, pp. 1306 – 1311, 2012.

[54] T. Piumsomboon, G. Lee, R. W. Lindeman, and M. Billinghurst, "Exploring natural eye-gaze-based interaction for immersive virtual reality," In *2017 IEEE Symposium on 3D User Interfaces (3DUI)*, pp. 36-39, 2017.

[55] J. B. Pelz, T. B. Kinsman, and K. M. Evans, "Analyzing complex gaze behavior in the natural world," In *SPIE-IS&T Human Vision and Electronic Imaging XVI*, vol. 7865, pp. 1-11, 2011.

[56] T. Pfeifffer and P. Renner, "Eyesee3d: A low-cost approach for analyzing mobile 3d eye tracking data using computer vision and augmented reality technology," In *Proceedings of the Symposium on Eye Tracking Research and Applications (ETRA)*, Safety Harbor, Florida, USA, 2014.

[57] R. Takahashi, H. Suzuki, J. Y. Chew, Y. Ohtake, Y. Nagai and K. Ohtomi, "A system for three-dimensional gaze fixation analysis using eye tracking

glasses," *Journal of Computational Design and Engineering*, Vol. 5, No. 4, pp. 449-457, 2018.

[58] R. R. Jensen, J. D. Stets, S. Suurmets, J. Clement, and H. Aanæs, "Wearable gaze trackers: Mapping visual attention in 3d," In *Scandinavian Conference on Image Analysis: 20th Scandinavian Conference (SCIA 2017)*, pp. 66-76, Springer volume 10269, 2017.

[59] L. Paletta, K. Santner, G. Fritz, A. Hofmann, G. Lodron, G. Thallinger, and H. Mayer, "Facts - a computer vision system for 3d recovery and semantic mapping of human factors," In *9th International Conference on Computer Vision Systems, ICVS 2013*, pp. 62-72, Springer volume 7963, 2013.

[60] T. Booth, S. Sridharan, V. Bethamcherla, and R. Bailey, "Gaze3d: Framework for gaze analysis on 3d reconstructed scenes," *Proceedings of the ACM Symposium on Applied Perception, SAP 2014*, 2014.

[61] J. Pieszala, G. Diaz, J. Pelz, J. Speir, and R. Bailey, "3d gaze point localization and visualization using lidar-based 3d reconstructions," In *Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications*, pp. 201-204, ACM, 2016.

[62] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91-110, 2004.

[63] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o(n) solution to the pnp problem," *International Journal of Computer Vision*, vol. 81, pp. 155-166, 2009.

[64] J. L. Schönberger, "Robust Methods for Accurate and Efficient 3D Modeling from Unstructured Imagery," Ph.D. thesis ETH Zurich, 2018.

[65] J. L Schönberger, F. Radenović, O. Chum, and J.-M. Frahm, "From single image query to detailed 3d reconstruction," In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5126-5134, 2015.

[66] Tobii Pro Glasses 2, https://www.tobiipro.com/product-listing/tobii-pro-glasses-2/. Accessed March 2020.

[67] B. Lu, X. Duan and Y. Yuan, "Facial expression recognition based on ensemble extreme learning machine with eye movements information," *Proceedings of ELM-2015*, Volume 2, pp. 295-306, Springer International Publishing, 2016.

[68] Tobii Pro Glasses 2 Eye Tracker Data Quality Test,

https://www.tobiipro.com/siteassets/tobii-pro/accuracy-and-precision-tests/tobii-pro-glasses-2-accuracy-and-precision-test-report.pdf

[69] COLMAP, https://colmap.github.io/. Accessed March 2020.

[70] J. L Schönberger and J.-M. Frahm, "Structure-from-motion revisited," In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4104-4113, Las Vegas, NV, USA, 2016.

[71] J. L Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm, "Pixelwise view selection for unstructured multi-view stereo," In *14th European Conference on Computer Vision (ECCV2016)*, vol. 9907, pp. 501-518, Amsterdam, The Netherlands, 2016.

[72] J. L. Schönberger, T. Price, T. Sattler, J.-M. Frahm, and M. Pollefeys, "A vote-and-verify strategy for fast spatial verification in image retrieval," In *13th Asian Conference on Computer Vision (ACCV2016)*, vol. 10111, pp. 321-337, Taipei, Taiwan, 2016.

[73] OpenMVS, http://cdcseacave.github.io/openMVS. Accessed October 2020.

[74] R. Hartley and A. Zisserman, "Multiple View Geometry in Computer Vision," Cambridge University Press, 2004.

[75] Y. Furukawa and C. Hernández, "Multi-View Stereo: A Tutorial," *Foundations and Trends® in Computer Graphics and Vision*, Vol. 9, No. 1-2, pp. 1-148, 2015.

[76] C. Barnes, E. Shechtman, A. Finkelstein and D. B. Goldman, "PatchMatch: A randomized correspondence algorithm for structural image editing," *ACM Transactions on Graphics (Proc. SIGGRAPH)*, vol. 28, no. 3, 2009.

[77] M. Jancosek and T. Pajdla, "Exploiting visibility information in surface reconstruction to preserve weakly supported surfaces," International Scholarly Research Notices, 2014.

[78] H.-H. Vu, P. Labatut, J.-P. Pons and R. Keriven, "High accuracy and visibility-consistent dense multiview stereo," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 5, pp. 889-901, May 2012.

[79] O. Faugeras and R. Keriven, "Variational principles, surface evolution, pde's, level set methods and the stereo problem," *IEEE Transactions on Image Processing*, vol. 7, pp. 336-344, 1998.

[80] M. Waechter, N. Moehrle and M. Goesele, "Let there be color! large-scale texturing of 3d reconstructions," In *Proceedings of 13th European*

Conference on Computer Vision (ECCV2014), pp. 836-850, Zurich, Switzerland, 2014.

[81] FFmpeg, https://www.ffmpeg.org/, Accessed March 2020.

[82] CGAL – 3D Fast Intersection and Distance Computation (AABB Tree), https://doc.cgal.org/latest/AABB_tree/index.html#title3

[83] D. S. Wooding, "Fixation maps: Quantifying eye-movement traces," In *Proceedings of the 2002 Symposium on Eye Tracking Research & Applications ETRA '02*, ACM, pp. 31-36, 2002.

[84] H. Song, J. Lee, T. J. Kim, K. H. Lee, B. Kim and J. Seo, "Gazedx: Interactive visual analytics framework for comparative gaze analysis with volumetric medical images," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 311-320, 2017.

[85] H. Song, J. Yun, B. Kim and J. Seo, "Gazevis: Interactive 3d gaze visualization for contiguous cross-sectional medical images," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, pp. 726-739, 2014.

Acknowledgements

Throughout the writing of this thesis, I have received a great deal of support and assistance.

First of all, I would like to extend my deepest gratitude to my academic supervisor, Professor Hiromasa Suzuki, for his supervision, support, patience, and encouragement. He provided me the opportunity to enter The University of Tokyo for my abroad study and has given me invaluable insights and suggestions. With his support, I was able to attend domestic and international conferences, and submit my research work to journals. Those experiences broadened my horizons. I shall never forget his supervision during this period.

I would like to express my sincere gratitude to Professor Hajime Asama, Professor Yusuke Tamura, Professor Seiichi Takamatsu, and Professor Yutaka Ohtake for their review of the thesis manuscript and the valuable advices that they provided. I definitely learned a lot from their comments and suggestions.

I would like to acknowledge Mr. Shinji Matsuda from Lattice Technology and Assistant Professor Tatsuya Yatagawa for their useful technical advice in the development of the AR-based assembly assistance system.

I also appreciate Secretary Rumi Tsujiguchi, Associate Professor Yukie Nagai, International Multidisciplinary Engineering (IME) office, my lab colleagues: Ryo Takahashi, Takeru Uchiyama, Shintaro Suzuki, and Yuki Doi, and my friend, Chien-Yu Lan, for their general campus help. In addition, I could not have completed this thesis without the support of my lab colleagues: Yu Wang, Jingda Mai, Yifan Yang, and Xiangning Mao, and my friend, Po-Hsun Chen, who provided stimulating discussions and happy distractions to rest my mind.

Finally, I would like to express thanks to my beloved parents and my older sister for their love, encouragement, and understanding throughout my life.

List of Publications

Journal

- Ting-Hao Li, Hiromasa Suzuki, and Yutaka Ohtake, "Visualization of user's attention on objects in 3D environment using only eye tracking glasses," *Journal of Computational Design and Engineering*, volume 7, issue 2, pp. 228-237, April 2020, https://doi.org/10.1093/jcde/qwaa019

Conference paper with review

- Ting-Hao Li, Hiromasa Suzuki, Yutaka Ohtake, Tatsuya Yatagawa, and Shinji Matsuda, "AR-based assembly assistance system with efficient evaluation of misalignment between virtual and real objects," International Conference on Applied Human Factors and Ergonomics (AHFE 2020), Advances in Usability, User Experience, Wearable and Assistive Technology, pp. 690-697, https://doi.org/10.1007/978-3-030-51828-8_91 (2020)

Oral presentation

- Ting-Hao Li, Hiromasa Suzuki, and Yutaka Ohtake, "Evaluation system of user's attention on products in 3D environment using eye tracking glasses," 2018 Asian Conference on Design and Digital Engineering (ACDDE 2018), Okinawa Zanpamisaki Royal Hotel, Yomitanson, Okinawa, Japan, Nov. 1-3, 2018.
- 李 庭豪, 鈴木 宏正, 大竹 豊, 谷田川 達也, 松田 紳二, HMD を用いた AR アプリケーションのための 3 次元位置合わせの実装，日本機械学会 第 29 回設計工学・システム部門講演会講演論文集，2302 (2019)
- 李 庭豪, 鈴木 宏正, 大竹 豊, 谷田川 達也, 松田 紳二, Efficient method to evaluate misalignment between virtual and real objects for AR-based assembly assistance system、精密工学会春季講演論文集, (2020)