Building Scale VR: Automatically Creating Indoor 3D Maps and its Application to Simulation of Disaster Situations

Katashi Nagao
Graduate School of Informatics
Nagoya University
Nagoya, Japan 464–8603
Email: nagao@i.nagoya-u.ac.jp

Yusuke Miyakawa
Graduate School of Informatics
Nagoya University
Nagoya, Japan 464–8603
Email: miyakawa@nagao.nuie.nagoya-u.ac.jp

***Abstract*—It is useful to simulate disaster situations by reconstructing actual buildings in a virtual space to enable people using the buildings to learn how to act in a disaster situation before it occurs. Therefore, we are developing a disaster-simulation system that simulates various disaster situations by virtually reproducing the situation inside buildings to allow individuals to experience disaster situations by using the latest virtual reality (VR) system. We use a mobile robot equipped with multiple laser-range sensors that measure the distance to objects in a building and an RGB-depth camera to collect distance and image data while the robot automatically travels along a route suitable for 3D measurement. We also manually scan physical objects individually by using a handheld 3D sensor. We then arrange the objects in a 3D map and manipulate them. We have also developed a VR system called "Building-Scale VR" that consists of indoor 3D maps filled with manipulable virtual objects that we call "operation targets" and a VR headset capable of position tracking within the building. In this paper, we explain how to implement Building-Scale VR and its applications to disaster simulations. It is useful to express disaster situations by reconstructing actual buildings into virtual space and enable users in the building to experience such situations beforehand to learn how to properly act during a disaster.**

***Keywords*—Virtual reality; 3D map; autonomous mobile robot; disaster simulation**

I. Introduction

It is generally very expensive to create indoor 3D maps. However, as most recent information needs to be reflected on maps used for various purposes such as advertising, it is necessary to have mechanisms in place that allow for easy editing and updating content in accordance with the requirements of facility administrators. Namely, a mechanism is needed to automatically collect data necessary for automatic 3D-map generation. In this study, we collected sensor data using a mobile robot called the "automatic room capturer (ARC)" that can freely travel along an arbitrary route in an environment.

The ARC is a small unmanned robot capable of autonomous driving that we are currently developing in our laboratory. The ARC generates a probabilistic occupancy-grid map of a 2D plane (probabilities of the existence of objects according to x-y coordinates) by traveling in the environment beforehand. It is also possible to estimate its current position and have it operate autonomously by using the grid map.

Methods have been proposed for 3D-map generation that use 3D light detection and ranging [1] and camera images [2]. However, these methods are not without fault. The former cannot use color information, which makes it difficult to generate a map that is easily understood by humans. The latter has issues in that it is not possible to accurately generate a 3D map in an environment with several dark places or few image-feature points. A method called RGB-depth iterative closest point (RGBD-ICP) uses an RGB-D camera and generates a precise 3D map by superimposing a 3D point cloud based on image-feature points [3].

However, as this method uses only the data of the RGB-D camera when generating a 3D map, it is necessary to calibrate the real-world positional information and 3D map coordinates. If the real-world positional information is already related to the virtual world via the use of a probabilistic occupancy-grid map (probabilities of the existence of objects according to x-y coordinates) and additionally generating a 3D map using the same coordinate system as the existing map, it is possible to generate a 3D map that accurately reflects real-world information. This is why we decided to use an RGB-D camera on the ARC and developed a mechanism to automatically collect data by generating a data-collection route based on the occupancy-grid map that also uses location information at the time of data collection. We then developed a method that extends the RGBD-ICP, which allows not only the use of 3D point clouds and image-feature points but also the grid map.

We are developing a disaster-simulation system that simulates various disaster situations by virtually reproducing such situations in buildings that enable people to experience them by using the latest virtual reality (VR) systems. This system is based on 3D-map technology.

Since it is not easy to edit and modify automatically generated indoor 3D maps due to the huge amount of data, we also manually scan 3D objects individually by using a handheld 3D sensor. We then arrange them on a 3D map and consider them as operation targets.

To properly simulate disaster situations, it is important to

model the moving or deforming of physical objects (which we call "operation targets") in accordance with physical laws. This is why we developed a VR system called "Building-Scale VR" that consists of indoor 3D maps with operation targets and a VR headset capable of position tracking within a building. We use a 2D identification marker called an "ArUco code" to automatically arrange the operation targets on 3D maps. The ArUco code functions as a unique ID when scanned and recognized as a 3D point cloud. It can be decoded to identify the position and orientation of a particular operation target.

Although the operation targets can be arranged in arbitrary positions on a 3D map, it is difficult to locate them appropriately. Therefore, we also developed a method of automatically determining the xyz-coordinates of operation targets on 3D maps according to their actual positions. To automatically arrange the operation targets on a 3D map, we use an ArUco code.

After arranging multiple ArUco codes at various places in an indoor environment, our automatic method creates a 3D map, and the operation targets are arranged in positions based on their corresponding IDs encoded in the ArUco codes. Therefore, the initial position and orientation of each operation target are automatically adjusted and arranged on the 3D map.

In the virtual space, not only the movements of objects but also those of people are simulated, enabling confirmation on whether evacuation can be carried out properly.

In this paper, we explain our automatic 3D-map-generation method, Building-Scale VR based on this method, then argue that Building-Scale VR is useful for disaster simulations.

II. Related Work

Disaster-simulation systems that use VR technology have been developed to improve the sense of reality of disaster experiences and for portability.

For example, Sinha et al. [4] made it possible to experience the actual characteristics of a building and furniture during an earthquake, which are calculated using the finite element method, in an immersive VR environment. In such systems, a 3D model of a room and objects are manually created. However, it is difficult to modify the created content to adapt it to another room. The effort required to create new content for another room is immense.

Various methods for measuring an environment using a camera or laser to create a 3D model have recently been developed. The simultaneous localization and mapping (SLAM) method estimates the position and direction of a camera and constructs a model of the sensing-target environment [5]. The KinectFusion method creates a dense 3D model of an environment using an RGB-D camera [6], enabling the inexpensive creation of 3D models of various environments in a short amount of time.

Segmentation methods have also been developed to extract individual objects from the 3D point-cloud model obtained through measuring the entire environment [7]. In particular, Jia et al. [8] and Zheng et al. [9] proposed such methods based on the laws of physics.

However, these methods carry out segmentation primarily for the purpose of robots to understand their environments and are not aimed at realistically representing the models of real objects. Thus, segmentation methods cannot be directly applied to disaster-simulation systems that use VR technology since the objects' 3D models' unnatural appearance negatively affects the realism of the experience.

In our study, we developed a mechanism to automatically generate highly realistic VR content based on a 3D point cloud using an autonomous mobile robot and automatically arranged 3D objects that were manually modeled individually. This enables the generation of VR content with high reusability without impairing the sense of reality. We also developed a mechanism for estimating indoor position and orientation to allow individuals to navigate through the corresponding VR environment from an arbitrary position in a real building. Based on the method of estimating the relative position and direction between 3D point-cloud models [10].

III. Automatic Room Capturer (ARC)

The ARC can recognize its surroundings, collect various types of data by exploring its environment, and use these data to provide advanced services. The ARC can also run autonomously and trace a generated route to sense every corner of an environment.

The appearance of the ARC is as shown in Fig. 1. It is configured with iRobot Create[1], which has a facing two-wheel type moving mechanism and is equipped with a PC, multiple laser-range sensors, and an RGB-D camera.

A laser-range sensor detects the distance from it to the object on a 2D plane. The ARC uses such sensors to generate a probabilistic occupancy-grid map and estimate its position.

It uses Microsoft Kinect[2] as the RGB-D camera. In addition to the camera, Kinect also has a depth sensor that can measure the distance to an object detected with the camera. Unlike a laser-range sensor, which can measure the distance to an obstacle on a plane, Kinect can recognize the 3D shape of an object.

The ARC must always detect its position on the grid map to ensure flexible autonomous driving along a designated route. The position is defined as the posture of the moving object in the coordinates of the map and the angle from the x-axis $\mathbf{a}_t = (x_t, y_t, \theta_t)$. Posture is a value determined by the position x, y and orientation θ of the robot. The ARC can estimate its position on the grid map by comparing the values of the map with those of the laser-range sensors in real time while autonomously operating along an arbitrary route. The ARC can perform self-localization with an accuracy of about a 7-cm error.

A. Occupancy Grid Map

A map of the driving environment is necessary to determine the current location and desired destination to allow autonomous travel of the ARC. The ARC creates a 2D occupancy-grid map from the measurements recorded with the

[1]iRobot Create is a trademark of iRobot Corporation.
[2]Microsoft Kinect is a trademark of Microsoft Corporation.

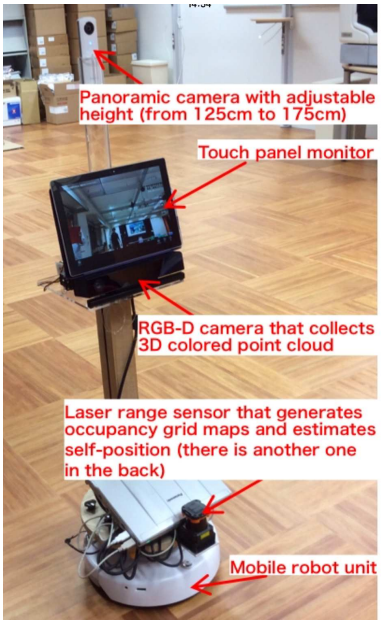

Fig. 1. Configuration of ARC.

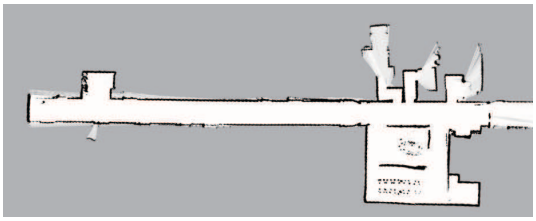

Fig. 2. Example of occupancy grid map.

laser-range sensors from previously traveling the environment. The occupancy-grid map is a map represented by a random variable located in an equidistant grid, which means that the higher the random variable, the higher the possibility that an area is occupied by an object.

An occupancy-grid map generated by the ARC is shown in Fig. 2. The color intensity of each pixel of the image represents the magnitude of the random variable of each grid of the occupancy-grid map. Black indicates an obstacle, white indicates no obstacle, and gray indicates an indeterminate state.

IV. Automatic Generation of Indoor 3D Maps

The indoor-3D-map generation carried out in this study consists of the following four steps:

1) Generation of the route for data collection.
2) Data collection.
3) Superimposition of sensor data.
4) Polygonization of planes in a 3D map.

Step (1) involves automatically generating a route on which the ARC operates when collecting data from the probabilistic occupancy-grid map. Step (2) involves operating the ARC along the route generated in Step (1) and recording the data from the RGB-D camera and positional information. Step (3) involves generating a 3D map by superimposing the collected data. Superimposition allows the creation of a 3D map correlated with the grid map that uses corresponding points from the 3D point cloud and image-feature points between frames and corresponding points of the grid map.

The 3D map generated by superimposing the sensor data is a set of a huge number of 3D points; thus, the data set is large and very difficult to handle. We compensate for this in Step (4) by focusing on planes, such as walls and floors, which are the main elements that comprise the interior of a building. We simplify the 3D map by automatically extracting the planes in the map and polygonizing them.

A. Generation of Route for Data Collection

It is ideal to use as many image-feature points as possible from the collected data to be used for superimposing the sensor data. In indoor settings, the characteristic areas are concentrated along the walls and are often not detected on the floor. Therefore, the ARC runs along the walls and generates a route for collecting data, such as the center of the passage. Since there may be a region where data cannot be obtained due to being physically obstructed by other objects along the generated route, the ARC goes around the route in the opposite direction after completing the first run-through to compensate for potentially missed data.

Specifically, we create reference nodes from the grid map and connect the nodes to create a graph structure. The data-collection route is then generated using the generated graph structure.

B. Generating Route Graphs

A node is generated based on the distance from a wall in the traversable area of the environmental map to generate a route along the wall. Fig. 3 shows how the traversable area is color-coded based on the distance from the wall. It becomes darker as it becomes closer to the wall and becomes light blue as it gets farther away. We used a k-d tree [11] to calculate the distance to the wall. We generated nodes at constant intervals on the boundary where the color changes. The node generated is node A and is denoted with a red dot in Fig. 3. In narrow passages where node A is not generated, nodes are created in the middle of these passages. If the distance to the nearest wall (called the “nearest point”) at a point within an arbitrary traversable region is short and the distance to the wall on the opposite side is equidistant from the nearest point, it is then judged that the nearest point will be the center of the passage. The center of a narrow passage is indicated with a red line in Fig. 3, and the node generated on the line is indicated with a green dot as node B. Node C is generated as a connection point between nodes A and B at a position where the distance from the wall is determined to be the center of the passage at a fixed distance. Node C is indicated with a yellow dot. The generated nodes are connected to each other to generate a graph structure. We define two types of relationships between nodes: adjacent and reachable. Nodes generated on the same boundary line (nodes A, B, and C) are regarded as adjacent; these nodes and their connections are indicated in Fig. 4 with a red line. Reachable nodes are indicated with a blue line and represent nodes within a certain distance from a node that is not an adjacent node with the requirement that there are no obstacles between reachable nodes.

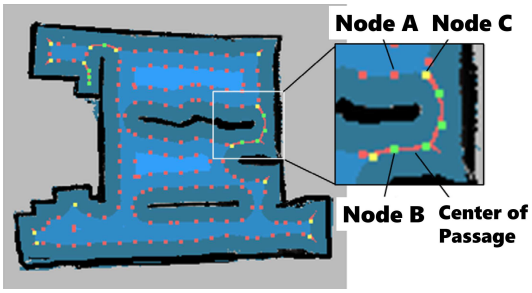

Fig. 3. State of node generation.

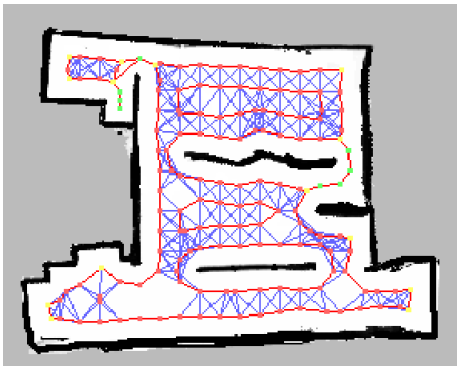

Fig. 4. Graph structure connecting nodes.

C. Route Generation for 3D Capture

The route for 3D capture of a room is generated in the following process:

Step 1 **Set the first node of the route to the nearest node (node A) from the current location.**

Step 2 **Generate a route** R **that goes around the adjacent nodes from a specified node** When a reachable node N_m from $\mathrm{R}_i (0 \leq i \leq n)$, which has already been searched, exists and node N_m has not yet been searched, add $\mathrm{R}_i \rightarrow \mathrm{N}_m$ to the search candidate. Also, delete the search candidate, which becomes $\mathrm{N}_j = \mathrm{R}_i$ if it exists.

Step 3 **Generate a route** R' **that passes through the unsearched node and insert it into route** R Choose the node with the shortest distance between the points in $\mathrm{R}_i \rightarrow \mathrm{N}_m$ from the search candidates and use node $\mathrm{N_m}$ as the starting point to execute the same operation as in Step 2 to generate route R'. Insert route R' in the $i+1$th node of route R. In this case, delete R'_0 to generate a smoother route if $\mathrm{R}_{i-1} \rightarrow \mathrm{R}'_1$ is reachable.

Step 4 **Repeat Step 3 until there are no search candidates.**

Step 5 **Perform Step 6 for all C nodes on the route.**

Step 6 **Insert route** R'' **going from node C to node B through a narrow space and insert in route** R Generate a route R'' going from node C to node B. The way of inserting route R'' into route R depends on node R''_n, which is the end of route R''. There are three cases of R''_n. Each case is explained below.

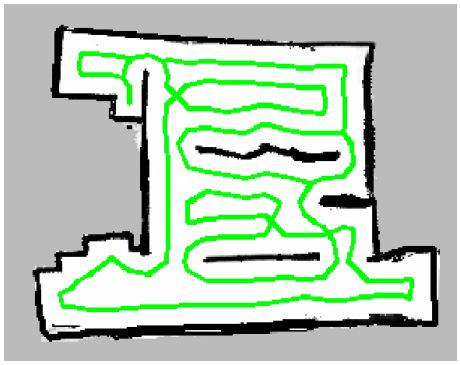

Fig. 5. Example of generated route.

1) The case in which R''_n is node B
We invert route R'', delete the 0th and nth nodes from it, then add it to the end of route R''. We then insert route R'' into the ith node of original route R.
2) The case in which R''_n is node C that is unsearched
Execute Steps 2-4 with R''_n as the starting point to generate route R'''. Invert route R'', delete its 0th and nth nodes, then add the inverted version to the end of route R'''. Delete the nth node of R'' and insert the deleted version at the beginning of route R'''. Then insert route R''' in the ith node of route R.
3) The case in which R''_n is node C that has already been found
Invert route R_{ij}, delete the 0th and nth nodes of route R'', then join them. Insert the result route at the jth position of R. Let the route from the ith node to the jth node of route R be route R_{ij} in which $\mathrm{R}_j = \mathrm{R}''_n$.

Step 7 **Add inversion of route** R **at the end of route** R**.**

The route generated in Fig. 5 was created using the above algorithm.

D. Superimposition of Sensor Data

It is possible to generate a 3D map associated with a real-world location by superimposing the sensor data using the correspondence relationship between the 3D point cloud and image-feature points between the frames and grid map.

It is possible to arrange the sensor data at positions corresponding to the grid map by using the position of the ARC as the initial position when superimposing the sensor data.

However, since the probabilistic occupancy-grid map is a 2D map, a 3D superimposition cannot be carried out. To superimpose the grid map and sensor data, 3D postures (positions and angles) of the ARC are required when data recording occurs. However, the posture data that the ARC can acquire are only x, y coordinates and the horizontal direction

(yaw θ_y). Since data are collected while moving, the sensors are not always fixed horizontally. Therefore, we try to detect the floor from the acquired sensor data, the z coordinate with the floor coinciding with the horizontal plane, and the rotation (pitch θ_p, roll θ_r) with respect to the vertical and traveling directions. The 3D position of the ARC at the time of data recording is then estimated.

E. Detection of Floor Plane

Detection of the floor enables us to assume the RGB-D camera levelness in comparison to the floor to some extent. The floor surface can be represented by the plane equation $ax+by+cz-1=0$. We use the random sample consensus (RANSAC) algorithm [12] to obtain the most floating parameters a, b, c. We select three points randomly from all the data and obtain a, b, c from those three points. At this point, if the surface, including the three points, is not horizontal to some extent, it is regarded as not being the floor. We calculate the error from the observed data by applying all the data to the obtained parameters and give rewards to the parameters if the error was within the allowable range. We repeat this process to extract data for the parameter with the largest amount of rewards and obtain another parameter from the extracted data to detect the most likely surface. In the event that the floor is not detected correctly due to the floor area not being included in the data, the data are not used for 3D superimposition. In this study, 1000 points sampled from sensor data were used as all the data, and the RANSAC algorithm was applied for 1000 iterations.

The 3D position is calculated using rotation matrix $\mathbf{R}$ and translation matrix $\mathbf{T}$ that converts a point p on the floor to point p' on the horizontal plane $p' = \mathbf{R} \cdot p + \mathbf{T}$. The $\mathbf{R}$ and $\mathbf{T}$ are obtained from three corresponding points between the horizontal plane and detected floor.

F. Method for Superimposition

To carry out superimposition, it is necessary to obtain the rigid body transformation matrix $\mathbf{A}$, which carries out rotation and translation. The $\mathbf{A}$ can be calculated from the position of the ARC by the following equation. Let $\mathbf{R}_x, \mathbf{R}_y, \mathbf{R}_z$ be the rotation matrix for each of the x, y, z axes, respectively.

$$\mathbf{A} = \mathbf{T} \cdot \mathbf{R}_z(\theta_y)\mathbf{R}_x(\theta_p)\mathbf{R}_y(\theta_r)$$

Since the position of the ARC was corrected so that the floor would be horizontal in this study, the z coordinate, pitch θ_p, and roll θ_r did not need to be modified. Therefore, only the x, y coordinates of the ARC and the left/right direction θ_y were corrected by superimposition.

Superimposition of 3D data is carried out with the following process:

Step 1 **Selection of Image-Feature Points** The image-feature points $\mathbf{F}_s$, points $\mathbf{M}_s$ used for matching with the grid map, and points $\mathbf{P}_s$ to be superimposed are extracted from the point cloud of the input frame. Points $\mathbf{F}_s$ are extracted based on the local features of the RGB image calculated using speeded up robust features (SURF) [13]. If a point corresponding to the extracted feature point in $\mathbf{F}_s$ does not exist in the point cloud, we ignore that point. For $\mathbf{M}_s$, we select the points where the transformed z coordinates of the points are around the height of the ARCs laser-range sensors. Points $\mathbf{P}_s$ are obtained by sampling from the point cloud. In this study, a quarter of the original point cloud was uniformly sampled in consideration of processing time and matching accuracy.

Step 2 **Selection of Frames for Matching** Of the several frames immediately before the input frame, the frames in which most $\mathbf{F}_s$ in the input frame matched are taken as the target frames for superimposition. When there is no matching frame, the frame just before the input frame is set as the target frame.

Step 3 **Matching and Weighting** We then find points corresponding to $\mathbf{F}_s$, $\mathbf{M}_s$, and $\mathbf{P}_s$ selected in Step 1, and let $\mathbf{S}_f$, $\mathbf{S}_m$, $\mathbf{S}_p$ be a set of each point. Set $\mathbf{S}_f$ is obtained by calculating the matching between the $\mathbf{F}_s$ of the input and target frames. If point f_t corresponding to point f_s does not exist, we delete f_s from $\mathbf{F}_s$.
The converted coordinates can be obtained with $\mathbf{A} \cdot p$, where p is on the sensor data. For $\mathbf{S}_m$, points m_t are selected if they are closest to points $\mathbf{A} \cdot m_s$, which are the points on the grid map converted to 3D coordinates.
For $\mathbf{S}_p$, points p_t are selected if they are closest to points $\mathbf{A} \cdot p_s$ extracted from the target frames.

Step 4 **Removal of Outliers** Since outliers adversely affect minimizing the error, we remove outliers from $\mathbf{S}_f$, $\mathbf{S}_m$, and $\mathbf{S}_p$. In $\mathbf{S}_f$, acceptable data are calculated using the RANSAC algorithm, and points not included in such data are removed as outliers. For $\mathbf{S}_m$ and $\mathbf{S}_p$, we remove points that are more than a certain distance as outliers. Also, in $\mathbf{S}_p$, if there are no data in the pixel adjacent to p_t on the depth image, we remove that point as the boundary of the 3D model.

Step 5 **Estimation of Errors and their Minimization**

Steps 3-5 are repeated until the amount of error is sufficiently small or the number of repetitions exceeds a predetermined limit.

G. Matching of 3D Map with Occupancy Grid Map

In addition to the correspondence of the 3D point cloud and $\mathbf{F}_s$ using the correspondence with the occupancy-grid map, superimposition of sensor data is used to generate a 3D map corresponding to the grid map. An example of a generated 3D map is shown in Fig. 6. To demonstrate the accuracy of the 3D map, we show the map of the data around the height of the ARC's laser-range sensors on a 3D map and 2D grid map in Fig. 7.

The red points indicate that the points on the 3D map matched those on the grid map. The generated 3D map was accurately associated with the grid map and the real-world location.

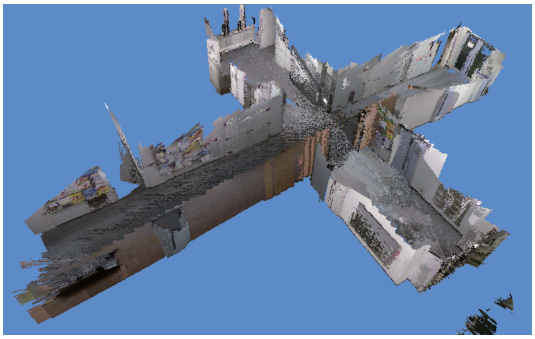

Fig. 6. Example of generated 3D map.

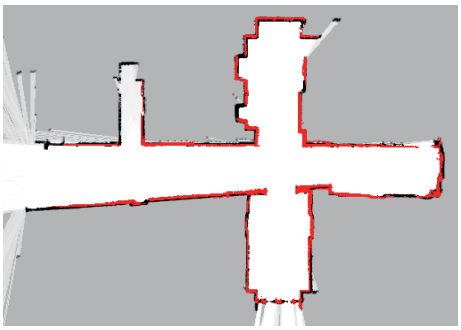

Fig. 7. Relating generated 3D map with its base grid map.

V. Operation Targets and Their Arrangement

Since 3D maps are represented as 3D point clouds, it is difficult to edit and modify them. Therefore, objects, such as movable furniture in a room, are individually scanned. A 3D model is then created and placed on a 3D map. We consider these 3D-modeled object as the "operation targets".

When creating a 3D map, we use a panoramic image captured using a 360-degree spherical imaging camera installed at the top of the ARC for coloring the 3D point cloud data. At the time of 3D data capture of a room, a 2D identification code is placed instead of an operation target. At the time of arrangement of each code, the orientation of the code is made to correspond to that of the operation target. As a result, a panoramic image including multiple 2D codes is created. By using this panorama image, the 2D codes are decoded and the placement position of the operation target is automatically determined.

A. ArUco Codes

From the created panorama image, the 3D map generation algorithm searches for the placed 2D codes. For this search, a marker-detection library, which is an OpenCV module of the image-processing library called ArUco, is used. Two-dimensional code that can be recognized with this library is called ArUco code. Examples are shown in Fig. 8. In ArUco, ArUco code is detected using a predefined dictionary, and its position and orientation are estimated from those of the camera, whereby the ID of the corresponding ArUco code and the 2D coordinates of the position in the image (Code_pos) [mm] and direction (Code_rot) are output. By transforming the

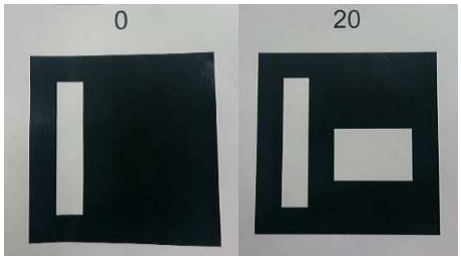

Fig. 8. Examples of ArUco codes.

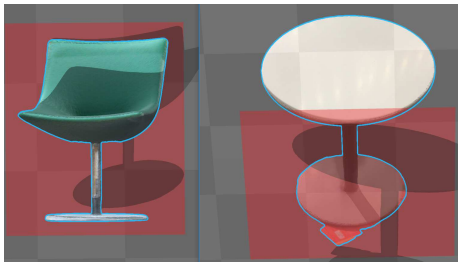

Fig. 9. 3D models of chair and table.

panoramic image into a sphere represented by 3D coordinates whose origin is the position of the camera, Code_pos and Code_rot are 2D coordinates converted into Code_pos 3D [mm] and Code_rot 3D of 3D coordinates.

Specific examples of ArUco codes are shown in Fig. 8.

B. Positioning of Operation Targets Based on ArUco Codes

To position the moveable/deformable objects (i.e., operation targets) in a 3D map as VR content, we must do the following first. We first generate 3D models by individually scanning the operation targets to be placed in the map by using a handheld 3D sensor. Example models are shown in Fig. 9.

Then, we associate the ID in the program of the operation target with that of the corresponding ArUco code. We use Unity for the programming environment of 3D graphics and animation.

After objects considered as operation targets are removed from the room and their corresponding ArUco codes are placed at each position, we then create a 3D map of the room. For example, in Fig. 10, one can see ArUco codes that have been placed in positions previously occupied by tables and chairs. A 3D map is subsequently created.

Three-dimensional models are placed on the 3D map based on the correspondence between the ArUco codes and operation targets. Fig. 11 shows a state in which the corresponding 3D models are automatically positioned at the location of the ArUco codes on the 3D map shown in Fig. 10.

For each panorama image, the 3D coordinates (Arc_pos) [mm] and orientation (Arc_rot) of the position of the ARC reference point, when the camera is shooting with the data-measurement starting point as the origin, are recorded. As a result, the position and orientation of the ArUco code are converted into coordinate axes with the data-measurement starting point as the origin. In addition, the 3D coordinates

Fig. 10. ArUco codes in the room.

Fig. 11. Replacement of ArUco codes with 3D models.

(Camera_pos) [mm] and orientation (Camera_rot) of the camera position with respect to the reference point of the ARC are also recorded. Camera_pos and Camera_rot are constants and used to correct the deviation between the ARC reference point and camera position. This makes it possible to calculate the position and orientation of the ArUco code in the room.

3D position of ArUco code in the room = Arc_pos + Camera_pos + Code_pos3D
3D orientation of ArUco code in the room = Arc_rot + Camera_rot + Code_rot3D

The 3D coordinates whose origin is the position of the panoramic camera and the 2D coordinates on the image plane have the following relationship.

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

where, (x, y, z) are the 3D coordinates whose origin is the camera position, (u, v) represents 2D coordinates on the image plane, A is a camera matrix, (c_x, c_y) represents the principal point, and (f_x, f_y) represents the focal length. Using this relational expression, 2D coordinates of the ArUco code on the image plane are converted into 3D coordinates whose origin is the camera position. In the actual camera, using the distortion coefficient represented by the vector $(k_1, k_2, p_1, p_2[, k_3[, k_4, k_5, k_6]])$, it is calculated as follows.

Fig. 12. User wearing HMD.

$$\begin{aligned}
x' &= x/z \\
y' &= y/z \\
x'' &= x' \frac{(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)}{(1 + k_4 r^2 + k_5 r^4 + k_6 r^6)} + 2p_1 x' y' + p_2(r^2 + 2x'^2) \\
y'' &= y' \frac{(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)}{(1 + k_4 r^2 + k_5 r^4 + k_6 r^6)} + p_1(r^2 + 2y'^2) + 2p_2 x' y' \\
r^2 &= x'^2 + y'^2 \\
u &= f_x \times x'' + c_x \\
v &= f_y \times y'' + c_y
\end{aligned}$$

VI. Building Scale VR for Disaster Simulation

A. VR Headset Capable of Position Tracking

It is necessary to determine the position on a 3D map from the current position in the building for Building-Scale VR. For outdoor use, GPS and electronic compasses are often used, but they do not work indoors. Therefore, we attach a compact RGB-D camera (for example, Google's Tango [14]) to a head-mounted display (HMD) and calculate the head position and orientation. This is achieved via a method called RGB-D SLAM [15] and not by the self-location-estimation method used by the ARC with laser-range sensors. This is due to the fact that the sensor attached to the human head moves in three dimensions, unlike those installed on mobile robots. It is possible to accurately estimate position and direction by matching the point cloud obtained from the 3D map and the head-mounted sensor.

Fig. 12 shows a user wearing an HMD with an RGB-D camera.

In this figure, the user has controllers in both hands. The controllers' postures are calculated based on the relative position from the RGB-D camera mounted on the HMD.

B. Disaster Simulation

It is generally difficult to carry out a preliminary exercise, such as an evacuation drill in a real building, to examine in detail the functional problems that can occur during a disaster and to plan a solution. This is why we decided to reproduce buildings in a detailed virtual world using a 3D map to simulate disasters, so that disaster-countermeasure planning would be more intuitive. Simulations are best carried out

Fig. 13. Simulated people walking on 3D map.

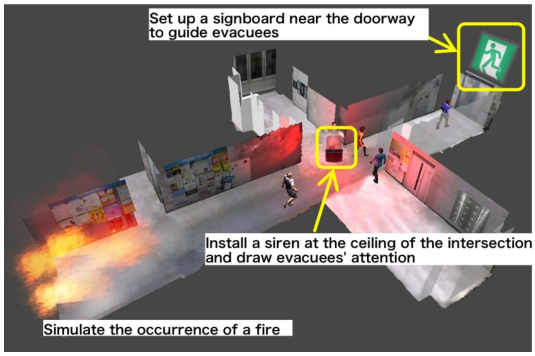

Fig. 14. Installing siren and signboard.

assuming various cases. However, there are limits to predicting in advance the types of disasters that can actually occur.

Therefore, we examined a disaster situation more intensively by placing objects, such as furniture, that move according to the laws of physics on an indoor 3D map. We also developed a method for intuitively reconfiguring a disaster situation based on a simple script applied to movable objects (i.e., operation targets) on the 3D map.

We implemented a disaster-simulation system that can easily enable the creation of effective disaster countermeasures for large-scale indoor facilities as an example application of Building-Scale VR.

1) Automatic Generation of VR Content Based on Indoor 3D Maps.
2) Simulation of Disaster Situations.
3) Collection of Human-Behavior Data.
4) Behavior-Learning and Disaster-Countermeasure Support.

Automatic Generation of VR Content Based on Indoor 3D Maps

As described above, using the automatic 3D-map-generation method with an autonomous mobile robot, a detailed map of a building is created. Furthermore, 3D maps can be converted to VR content, and 3D objects that are separately scanned can be automatically placed at appropriate positions. It is also possible to wander freely within a 3D map by wearing an HMD that can estimate the current position and orientation in real time.

Simulation of Disaster Situations

By using an indoor 3D map of a large-scale facility as a stage of multiplayer participation VR, as shown in Fig. 13, a 3D graphic object of a person walks around in the 3D map by using a behavior script (program of movement-pattern rules). Behavior scripts that can be easily created, modified, and applied to human 3D models generate animation of multiple human motions.

As shown in Fig. 14, we also developed a mechanism capable of generating a fire at an arbitrary place in a building represented by a 3D map and moving and deforming the objects in the building.

With this simulation system, it is possible to acquire as much information as possible before installing actual facilities for a disaster in a building. As shown in Fig. 14, for example, by installing a siren or a signboard at an appropriate place in a 3D map and devising methods to notify people in the facility of disaster information, we can evaluate how these methods can enable efficient evacuation.

Therefore, it is possible to arrange various objects on an indoor 3D map and define human-behavior patterns by using scripts. By creating scripts on how humans behave and react, it is possible to take measures against evacuation guidance by taking into account how sirens and signboards affect humans. We can incorporate various evacuation procedures and allow users to experience and evaluate them. Various types of disasters are conceivable, and structural problems during a disaster in real facilities can be clarified.

Collection of Human-Behavior Data

The system can record a user's behavior (such as how he/she moved) in the simulation of a disaster situation, and save it for each user. This is used not only for evaluating the reproducibility of the simulation but also for measuring the effect of additional virtual equipment described below.

Behavior-Learning and Disaster-Countermeasure Support

As described above, an indoor simulation system of a large-scale facility virtualizing a real building is made available to an unspecified number of users. This is to allow the administrator of the actual facility to disclose the system to these users so that it can be used at arbitrary times. A user can learn not only about the virtual disaster situations indoors but also how to act in such a situation in the real world. The facility administrator should add various tricks to the environment side, encourage users to voluntarily move and evacuate safely, and appropriately design the equipment.

VII. Future Plan

We will verify the effectiveness of Building-Scale VR by carrying out a demonstration experiment using the disaster-simulation system we are developing. Specifically, we will create a 3D map of the entire floor of a building (e.g. library) at Nagoya University and automatically arrange 3D objects that can be moved and deformed using ArUco codes. Next, using the disaster-simulation system, we will have virtual people

walk around on a 3D map by using behavior scripts. We will attempt to give an unspecified number of users a virtual experience of disaster evacuation and collect their action log data. By analyzing the log data, it will be possible to obtain information concerning evacuation guidance and building-interior redesign.

VIII. Concluding Remarks

We developed Building-Scale VR that consists of indoor 3D maps with operation targets and a VR headset capable of position tracking in a building. We developed a method for automatically measuring the inside of large facilities to generate 3D indoor maps. This is accomplished using an autonomous mobile robot that collects 3D data throughout a building, allowing the data to be accumulated and integrated. Building-Scale VR allows users to move within a 3D map by walking around a real building and at the same time making it possible to be immersed in a virtual version of the real world. We installed an RGB-D camera onto a regular VR headset allowing sensor data to be matched with map data to enable position tracking.

As Building-Scale VR allows virtual manipulation of objects in an actual building, it is suitable for simulating situations such as disasters. Our disaster-simulation system based on Building-Scale VR enables users to virtually experience disaster situations and provides feedback on actual evacuation behavior. This makes it possible to carry out complicated large-scale simulations and training, which have been very difficult.

Our disaster-simulation system has not yet been subjectively evaluated, but we believe it will be useful in disaster-countermeasure planning.

Acknowledgments

This work is technically supported by Business Development Department of Kozo Keikaku Engineering Inc. In particular, the authors are grateful to Yuki Matsuyama, Masayuki Umeda, Eiji Miyagaki and Masaki Tamada.

References

[1] S. Thrun, S. Thayer, W. Whittaker, C. Baker, W. Burgard, D. Ferguson, D. Hahnel, D. Montemerlo, A. Morris, Z. Omohundro, C. Reverte, and W. Whittaker, "Autonomous exploration and mapping of abandoned mines," *IEEE Robotics and Automation Magazine*, vol. 11, no. 4, pp. 79–91, 2004.

[2] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-time single camera SLAM," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 6, pp. 1052–1067, 2007.

[3] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments," in *Proc. of the International Symposium on Experimental Robotics (ISER)*, 2010.

[4] R. Sinha, A. Sapre, A. Patil, A. Singhvi, M. Sathe, and V. Rathi, "Earthquake disaster simulation in immersive 3D environment," in *Proc. of World Conference on Earthquake Engineering*, 2012.

[5] R. Newcombe and A. Davison, "Live dense reconstruction with a single moving camera," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 1498–1505.

[6] O. Kahler, V. Prisacariu, C. Ren, X. Sun, P. Torr, and D. Murray, "Very high frame rate volumetric integration of depth images on mobile device," *IEEE Transactions on Visualization and Computer Graphics, Vol.12, No.11,*, vol. 12, no. 11, pp. 1241–1250, 2015.

[7] A. Nguyen and B. Le, "3D point cloud segmentation: A survey," in *Proc. of IEEE Conference on Robotics, Automation and Mechatronics*, 2013, pp. 225–230.

[8] Z. Jia, A. Gallagher, A. Saxena, and T. Chen, "3D-based reasoning with blocks, support, and stability," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 1–8.

[9] B. Zheng, Y. Zhao, J. Yu, K. Ikeuchi, and S. Zhu, "Detecting potential falling objects by inferring human action and natual disturbance," in *Proc. of IEEE Conference on Robotics and Automation*, 2014, pp. 3417–3424.

[10] R. Rusu, N. Blodow, Z. Marton, and M. Beetz, "Aligning point cloud views using persistent feature histograms," in *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008, pp. 3384–3391.

[11] J. L. Bentley, "K-d trees for semidynamic point sets," in *Proc. of the Sixth Annual Symposium on Computational Geometry (SCG '90)*, 1990, pp. 187–197.

[12] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, pp. 381–395, 1981.

[13] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Speeded-Up Robust Features (SURF)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[14] D. Keralia, K. K. Vyas, and K. Deulkar, "Google Project Tango - A convenient 3D modeling device," *International Journal of Current Engineering and Technology*, vol. 4, no. 5, pp. 3139–3142, 2014.

[15] N. Engelhard, F. Endres, J. Hess, J. Sturm, and W. Burgard, "Real-time 3D visual SLAM with a hand-held camera," in *Proc. of the RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum*, 2011.