IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 10 November 2022, accepted 11 December 2022, date of publication 15 December 2022,
date of current version 21 December 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3229864

RESEARCH ARTICLE

Human–Robot Labeling Framework to Construct Multitype Real-World Datasets

AHMED ELSHARKAWY AND MUN SANG KIM, (Member, IEEE)
Center for Healthcare Robotics, School of Integrated Technology, Gwangju Institute of Science and Technology, Gwangju 61005, South Korea

Corresponding author: Mun Sang Kim (munsang@gist.ac.kr)

This work was supported in part by the Ministry of Trade, Industry and Energy of Korea under Grant 20003762; in part by the Open AI Dataset Project (AI-Hub, South Korea), in 2022; and in part by the Gwangju Institute of Science and Technology (GIST) Research Project Grant funded by the GIST, in 2022.

ABSTRACT The rapid development of deep learning object detection models opens great chances to build novel robotics applications. Nevertheless, several flaws are observed when deploying state-of-art object detection models with robots in a real-world environment, attributable to the discrepancy between the robots' actual observed environments and training data. In this study, we propose a labeling framework that enables a human to guide a robot in creating multitype datasets for objects in the robot's surroundings. Our labeling framework ensures no usage of labeling tools (e.g., software) but a direct hand-free gesture-based interaction between humans and robots. Using our labeling framework, we can enormously reduce the effort and time required to collect and label two-dimensional and three-dimensional data. Our system was implemented using a single RGB-D sensor to interact with a mobile robot, position feature points for labeling, and track the mobile robot's movement. Several robot operating system nodes were designed to allow a compact structure for our labeling framework. We assessed different components in our framework, demonstrating its effectiveness in generating quality real-world labeled data for color images and point clouds. It also reveals how our framework could be used in solving object detection problems for mobile robots. Moreover, to evaluate our system considering human factors, we conducted a user study, where participants compared our framework and conventional labeling methods. The results show several significant enhancements for the usability factors and confirm our framework's suitability to help a regular user build custom knowledge for mobile robots effortlessly.

INDEX TERMS Annotation tool, human–robot interaction, object detection, computer vision.

I. INTRODUCTION

The development of robot platforms that interact naturally with their surroundings has been a significant area of robotics research for decades. Thus, enabling robots to detect objects precisely near their reach is necessary to perform different tasks. The recent advancements in designing deep learning models have boosted the performance of object detection [1]. Unlike the superior performance of the state-of-the-art deep learning models to detect objects in a controlled environment, there is poor performance in a cluttered environment where objects of various sizes and shapes exist [2]. The nature of databases used in training the deep learning models is a fundamental cause of this contrast in performance [3], [4]. Using mobile service robots indoors (e.g., in homes, hospitals, and offices) requires training object detection models relative to items in each space. Reflecting on how items exist in a real-human environment must be considered when creating a dataset that best represents a robot's application situation (e.g., a mobile robot may approach an item from different viewpoints inside a cluttered environment) [5], [6]. Personalizing a detected item or connecting the detected item to a specific location in space (e.g., detecting a backpack that belongs to Yaseen or finding a glucometer that is usually placed in a medical cabinet) reflects a deep awareness of the robot's surroundings [7]. Finding publicly available datasets that serve this more profound knowledge is rare. Several publicly accessible datasets do not satisfy the previous

The associate editor coordinating the review of this manuscript and approving it for publication was Li He.

criteria, although they provide large datasets for general training classes.

The procedure to create an accurately labeled dataset comprises two main parts: the collection of raw data (e.g., color images, and point clouds) and the preparation of the collected data for training. Assuming quality raw data are collected, the tedious data preparation task requires careful handling, considering that data preparation takes up to 80% of every data science project's time [8], [9]. Issues such as equal class representation, quality labeling, data reduction, and cleansing are essential when preparing training data to receive a decent result from the deep learning models. Because data annotation is a crucial component of data preparation, many annotation tools have been developed to assist annotators in producing higher-quality work. However, existing tools necessitate human labor, are inaccurate when an inexperienced annotator is involved, and require a long working time [10]. Performing the same annotation task for two-dimensional (2D) data on three-dimensional (3D) data (e.g., point clouds) significantly increases task complexity [11].

Therefore, it is essential to think of a solution to eliminate the influence of training datasets that do not help a robot accurately detect objects in actual operation space and design an intuitive labeling method that helps users acquire and label 2D and 3D data effortlessly. We set a five-step guideline to search for possible solutions.

1) Reduce the burden of using labeling tools;
2) Equalize the effort to label 2D and 3D data;
3) User convenience must be considered;
4) A user can customize datasets to allow a robot to perceive an in-depth knowledge of an environment;
5) The resultant labeled data must be quality and reflect the operational environment settings of the robot.

As indoor service robots are exposed to a limited set of objects, exploring the possibility of teaching the robots about their working environment is a viable idea. Therefore, in this study, we design a labeling framework that teaches robots about objects in their working space by elaborating on the collaboration between a human and a mobile robot to customize the training datasets to optimally benefit the mobile robot's application situation.

For our labeling framework, we used a mobile robot equipped with an RGB-D camera, where we could track the camera pose using a simultaneous localization and mapping (SLAM) algorithm. The initial phase of our labeling framework starts with using the MediaPipe machine learning pipeline [12] to provide the fundamental 2D tracking for hand landmarks. Then, the 2D tracking information is transformed into a 3D position for the hand landmarks represented in the color camera frame. Afterward, the user can use dual hands directly to input six 3D feature points representing the approximate dimension of the object of interest. Notably, the user only needs to engage with the object of interest once and is not required to employ an input device, either wearable or handheld. Moreover, the user does not have to use any graphical user interface throughout the process. After delivering the feature points, we reflect the pose of the formed virtual bounding box by tracking the camera pose. Once the mobile robot reports a stop situation between movements, we simultaneously save the relative labeled color images, labeled point clouds, and label-related information files. This framework is designed to achieve minimal human intervention in the labeling process. Moreover, the structure of our custom-created datasets addresses several flaws in the publicly available datasets targeting applications of mobile robots.

The major contributions of this study can be summarized as follows.

- We proposed a labeling framework that mainly serves mobile robots to solve problems related to object detection in a real-human environment by filling the gap between training data and real-world settings.
- The labeling framework creates a direct collaboration between humans and mobile robots to accomplish the labeling process. This framework ensures minimal human intervention as it does not require the user to use software or physical tools to create or refine labels.
- To the best of our knowledge, this is the first real-time labeling framework that simultaneously creates multipurpose training datasets that depict the real-world mobile robot's application situation. Thus, the generated datasets could be processed to train various deep-learning models.

The remainder of this article is organized as follows. In Section II, we present related work. Section III describes the various design aspects of our labeling framework. Section IV shows the experimental setups and evaluation for different parts of the labeling framework; this section also includes a user study by inviting human participants to use and evaluate the designed framework. Section V presents the discussion, highlighting study outcomes. Finally, Section VI concludes this study and presents future work.

II. RELATED WORKS

In this section, we show the alternatives of real-world training data. Then, we present annotation tools for labeling color images and point clouds.

A. ALTERNATIVES OF REAL-WORLD DATA

The availability of quality annotated data is usually a keystone to developing deep learning models. Considering the cost-effectiveness when preparing training data, synthetic data are well-known substitution candidates for real-world data [13], [14]. Rendering pipelines, fusion models, and genetic algorithm networks can all be used to generate synthetic data. Synthetic data are helpful if real-world data collection is impossible or when associated with security or privacy issues. However, using synthetic data requires the availability of graphically accurate representations of real-world situations. Moreover, synthetic data are subject to misrepresentation or incompleteness [15], [16] and require expert human time and effort in designing and rendering. To overcome these limitations, efforts to increase the realism of synthetic data have successfully boosted the deep learning model performance [17], [18], [19]. Nevertheless, increasing the realism of synthetic data is a costly process.

At present, robots equipped with cutting-edge sensing apparatus can create a digital replica of the real environment [20], [21]. As real-world collected training data produces excellent detection results, optimizing datasets to match what robots see in the real environment is beneficial. Besides, using synthesized data could be useful in several cases. However, if human–robot collaboration can successfully acquire and annotate real-world data, it must be considered a potential channel to train robots for their specific applications. To this extent, we seek to eliminate inaccuracy due to differences between training and real-world data using situation-specific collected and labeled data to prepare training datasets.

B. ANNOTATION TOOLS

Notably, the number of annotation tools for 2D images is increasing [22]. To facilitate the labeling task, some tools provide the ability to load an artificial intelligent (AI) model [23] (e.g., COCO SSD [24]) so it can identify and set initial labels for objects in an image. This feature is helpful, but the identified object must have a predefined class in the dataset used to train the AI model. Nevertheless, initial identifications are provided; a human user must double-check the accuracy of the given labels, if any, and then resume labeling the remaining objects in the scene. In general, the annotation of color images is manageable due to the available tools' maturity and ease of handling.

Although numerous annotation tools exist to label color images, there are not as many generic options available to annotate point clouds. Further, a prominent portion of 3D annotation tools are designed to label LiDAR-collected data to train models for autonomous driving [25], [26], [27]. Nguyen et al. [28] presented an interactive 3D–2D tool for segmenting and annotating data collected using an RGB-D sensor. This tool, like Wong et al.'s [29] annotating tool, works to annotate real-world data. The data must be collected beforehand for both annotation tools. Besides, the user must be familiar with using each tool's functions. As point cloud manipulation is necessary to place an appropriate 3D bounding box, it is less intuitive to perform such a task on a 2D screen. In [30], a virtual reality tool was developed to annotate point cloud data in a virtual 3D environment. When annotating 3D data in a 3D environment, the user is more immersed, but additional hardware (e.g., controllers and head-mounted display) adds to the system's complexity. All previously mentioned custom-built 3D annotation tools follow the conventional annotation pipeline in their design. That pipeline starts with collecting the raw data or utilizing the publicly available datasets and then performing filtering or reconstruction if needed. To accelerate the annotation process, some tools provide semiautomatic labeling functions that indeed demand human intervention to check the label's validity. Lastly, the user should learn the functions provided by each annotation tool, spending a fair amount of time learning and placing the labels. This whole effort of designing annotation tools is valuable but requires expert users' presence to guarantee the annotation success. Therefore, the techniques available to label point clouds still need to be sufficiently mature to provide an intuitive labeling experience and simplify the complexity of handling 3D data.

An alternative solution to avoid data labeling complexity is requesting the services of commercial companies like BasicAI [31], Scale [32], and Playment AI data solutions [33]. Outsourcing data labeling to commercial companies speeds up the time for big-scale projects. However, doing so is expensive and risks labeling accuracy when less-skilled employees are engaged. Moreover, a crucial flaw when using cloud-based annotation tools is the lack of security when annotating sensitive information.

Consider a mobile robot trying to detect its surroundings. Some objects may not be represented as classes in the dataset used to train the AI model employed by the robot, or they may have representation classes, but their shapes and sizes are unique. In such a case, the robot's operator needs to add new classes to the dataset. The operator should follow the earlier-mentioned annotation pipeline selecting one or more annotation tools that will consume considerable time and effort. Alternative techniques should be developed to meet the same task without adhering to the conventional labeling pipeline, exploiting the robot's presence. Thus, Gregorio et al. [34] demonstrated a semiautomatic labeling technique facilitating interaction between a human and an arm robot. This technique aims to label data acquired from real-world settings for a robot application domain. Despite the low effort they achieved in collecting industrial and daily-living related datasets for small objects, their system's user should hold a labeling tool and interact with software for label refinement. Moreover, the proposed technique can work with small objects that can fit under a robot arm's workspace but cannot serve the same purpose for larger objects placed far from the robot arm. In addition, this technique facilitates the labeling for 2D data (color images) only.

Targeting most of the stated drawbacks, we design our labeling framework by exploring the difficulties facing available annotating tools and trying to resolve them. Thus, our labeling framework delivers a mobile robot with the annotated 2D and 3D data for their real operation environment by involving both the user and robot in the labeling loop.

III. SYSTEM DESIGN

In this section, we describe several components of our labeling framework. The implementation comprises three stages: the initial labeling stage, the camera tracking stage for iterative labeling, and the dataset-construction stage for 2D- and 3D-labeled data.

A. HARDWARE AND OPERATING SYSTEM

In this study, we used the Turtlebot3 waffle-pi robot [35] to represent a basic form of a mobile robot. The robot is

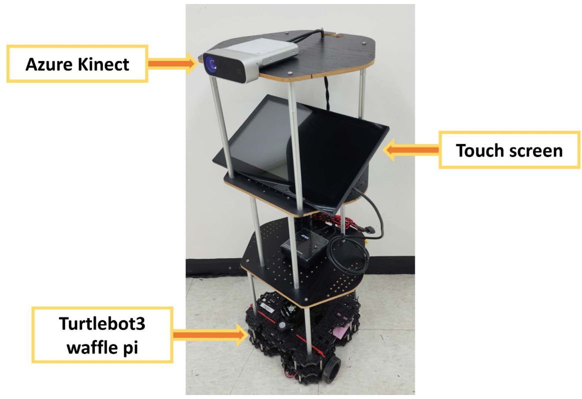


FIGURE 1. Mobile robot platform equipped with an RGB-D camera.

equipped with a Microsoft Azure Kinect camera [36] to stream color and depth images and a touch screen to provide a user with visual feedback (Fig. 1). The mobile robot is a modified version of the one we previously used in [37]. The operating system for the central processing unit (CPU) was Ubuntu 18.04LTS with the robot operating system (ROS) melodic environment. All supportive drivers and ROS packages were installed on the CPU.

B. INITIAL LABELING STAGE

This stage presents how a human user interacts with the mobile robot to assign feature points. The process of rendering the label bounding box is also illustrated in this stage.

1) INTERACTION INPUT MODALITIES AND 3D TRACKING OF THE FEATURE POINT

At this point, the robot faces both the user and object. The Microsoft Azure Kinect camera detects and tracks the landmarks on the user's two hands. We used the fundamental features of the MediaPipe machine learning pipeline for detecting and tracking the user's hand landmarks. We accessed the tracking landmarks for two hands and created a hand gesture identifier algorithm so the user could pass commands and information to the robot. According to the flowchart in Fig. 2, the user can use the right hand to make the primary gesture (e.g., pointing, canceling, and start scanning) and the left hand to perform a confirmation gesture, which is a closed palm gesture. The system performs the primary gesture-related operation after the confirmation gesture is detected (Fig. 3). This dual hand gesture identifier algorithm makes it easy for the user to communicate with the robot.

Because our primary goal is to label objects in their real 3D environment, it was essential to store the 3D information of the hand landmarks (e.g., capture the 3D coordinate of the index fingertip point). This goal was achievable by tracking a designated hand landmark in a color image to find the pixel coordinates and then searching for the corresponding depth information of that landmark. Then, the two pieces of information were transformed into a 3D coordinate for the target landmark represented in a selectable coordinate frame (e.g., color camera or depth camera coordinate frames; Fig. 3).

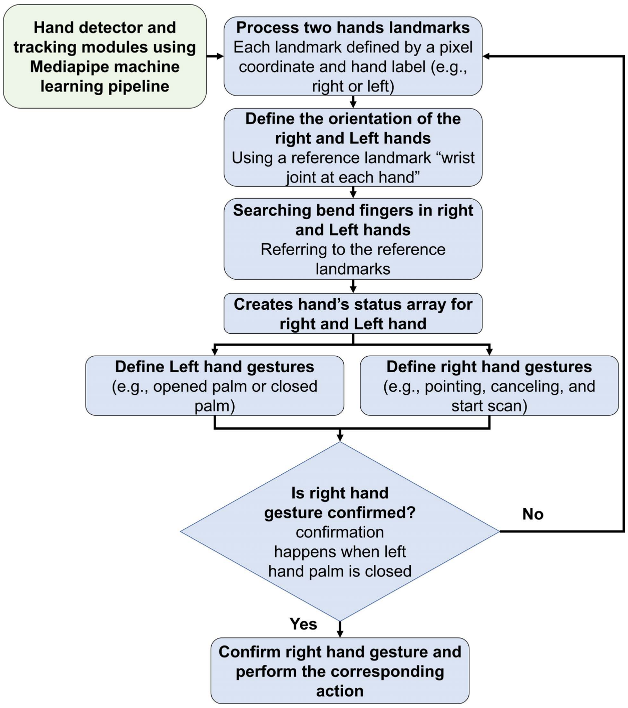

Hand detector and tracking modules using Mediapipe machine learning pipeline
Process two hands landmarks
Each landmark defined by a pixel coordinate and hand label (e.g., right or left)
Define the orientation of the right and Left hands
Using a reference landmark "wrist joint at each hand"
Searching bend fingers in right and Left hands
Referring to the reference landmarks
Creates hand's status array for right and Left hand
Define Left hand gestures
(e.g., opened palm or closed palm)
Define right hand gestures
(e.g., pointing, canceling, and start scan)
Is right hand gesture confirmed?
confirmation happens when left hand palm is closed
No
Yes
Confirm right hand gesture and perform the corresponding action

FIGURE 2. Dual hand gesture identifier algorithm.

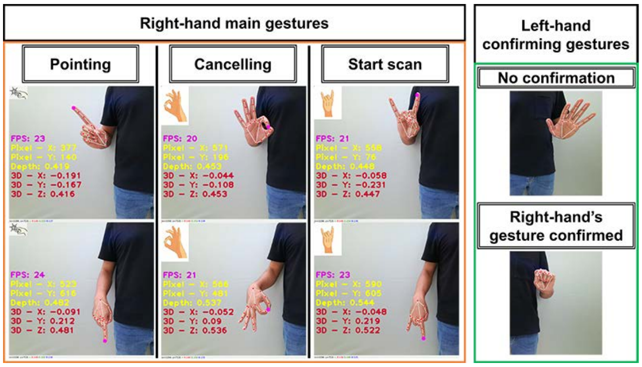


FIGURE 3. Detecting hand gestures in various orientations and tracking index fingertip 3D position.

Using our framework, the user registers several feature points for an object to create its label. Each feature point is a 3D position coordinate of the user's right hand index fingertip. Feature points are used to generate a virtual cuboid bounding box around the object of interest.

To input the label name, the user can speak the object name into the Azure Kinect camera's microphone, and if needed, he/she can add information such as to whom this object belongs and where it is usually seen or placed. As a standard for our work, we followed this sequence: the user speaks the word "start" to enable the registration of the label name and then speaks the object name (e.g., Multimeter). If the user needs to add information about the object owner, he/she says "space" as a separate word and then registers the owner's name (e.g., Park). Finally, to register the usual place to store or see the object, the user speaks the word "space" followed by the location name (e.g., Room 104). At any time, the user can terminate the registration for the label name by speaking the word "end."

We can input the object's label name during the initial labeling stage using the Microsoft Azure speech recognition application programming interface (API) [38], [39]. The reason we selected the Microsoft Azure API is its superior performance compared with other speech recognition APIs (e.g., Google API, and Amazon API). We have previously accomplished an exhaustive analysis of the use of cloud-based speech recognition APIs to select the best for integration with a home-developed sentence-level visual–speech recognition deep learning model [29], [30].

2) RENDERING INITIAL 3D BOUNDING BOX

The user interacts with the object of interest by assigning six feature points to the physical object's body. To assign a feature point, the user makes a pointing gesture with the right hand, placing the index fingertip above the desired point to register its coordinates. Four of the six points are mandatory for cuboid drawing ($\mathbf{P_0}$, $\mathbf{P_1}$, $\mathbf{P_2}$, and $\mathbf{P_3}$), and the other two are auxiliary points ($\mathbf{Aux_1}$ and $\mathbf{Aux_2}$) to enhance the accuracy of the virtual bounding box (Fig. 4). The mandatory feature points are placed over the object's corners, whereas the auxiliary points are placed between points $\mathbf{P_1}$ and $\mathbf{P_2}$.

To ensure the cleanness of registered feature points and the accuracy of the generated bounding box, we performed a check procedure comprising two steps.

1) We saved 20 records for each input feature point to search for and discard possible outliers. Then, the average value was calculated for the remaining position coordinates. This step aims to purify the depth values because they are more likely to show inaccurate measurements.
2) Assuming that frame {B} origin point is located at the center of the generated bounding box and frame {C} represents the color camera coordinates frame, we double-check the relative orientation between frames {B} and {C}. Thus, we calculated the angle between the X-axis of frame {C} and the vector formed between points P_1 and P_2. Similarly, we repeated the process to find the angle between X-axis of frame {C} and the vector formed between $\mathbf{Aux_1}$ and $\mathbf{Aux_2}$. We compared the two resultant angles and allowed a tolerance of 3°. When the difference was less than the tolerance margin, we accepted the measurements and calculated the average value between the two angles. Otherwise, we rejected the measurements and asked the user to re-enter the feature points.

Notably, asking the user to re-enter feature points was uncommon during operation. However, it was necessary to include the check steps to guarantee the quality of the feature points.

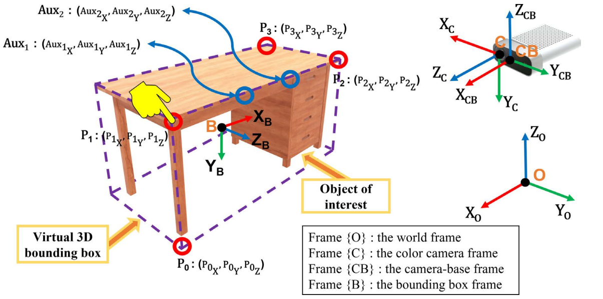

Aux₂ : (Aux2X, Aux2Y, Aux2Z)
Aux₁ : (Aux1X, Aux1Y, Aux1Z)
P₃ : (P3X, P3Y, P3Z)
P₂ : (P2X, P2Y, P2Z)
P₁ : (P1X, P1Y, P1Z)
P₀ : (P0X, P0Y, P0Z)
B
XB
ZB
YB
Object of interest
Virtual 3D bounding box
XC
ZCB
C
CB
ZC
YCB
XCB
YC
ZO
O
XO
YO
Frame {O} : the world frame
Frame {C} : the color camera frame
Frame {CB} : the camera-base frame
Frame {B} : the bounding box frame

FIGURE 4. Positions to assign feature points and coordinate frames.

After accepting the feature points' quality, we could determine the dimensions of the cuboid shape bounding box, as in (1). These dimensions are a close representation of the object of interest's actual dimensions if it is depicted as a cuboid. Referring to "(2)," we can calculate frame {B} origin point's coordinates ($B_{O-X}, B_{O-Y}, B_{O-Z}$) relative to the color camera frame {C}. The origin point's coordinates represent the relative displacement between the two frames, as in (3). The rotation of the generated bounding box relative to the camera frame {C} can be driven as in (4). Finally, the pose of frame {B} relative to frame {C} is a combination of the linear displacement and rotation components following (5).

$$\begin{aligned}&\text{Bounding box length}(L) \;= |\mathbf{P_1P_2}|\\&\text{Bounding box width}(W) \;= |\mathbf{P_2P_3}|\\&\text{Bounding box height }(H) = |\mathbf{P_0P_1}|\end{aligned} \tag{1}$$

$$\begin{aligned}B_{O-X} &= (P_{0_X} + P_{3_X})/2\\B_{O-Y} &= (P_{0_Y} + (\frac{P_{1_Y} + P_{2_Y} + P_{3_Y} + Aux_{1_Y} + Aux_{2_Y}}{5}))/2\\B_{O-Z} &= (P_{0_Z} + P_{3_Z})/2\end{aligned} \tag{2}$$

$$\text{Relative displacement } \left(\mathbf{D}_{\{B\}}^{\{C\}}\right) = \begin{bmatrix} B_{O-X} \\ B_{O-Y} \\ B_{O-Z} \\ 1 \end{bmatrix} \tag{3}$$

$$\text{Relative orientation } \left(\mathbf{R}_{\{B\}}^{\{C\}}\right) = \begin{bmatrix} \left(\frac{\mathbf{P_2-P_1}}{L}\right)' & \left(\left(\frac{\mathbf{P_3-P_2}}{w}\right) \times \left(\frac{\mathbf{P_2-P_1}}{L}\right)\right)' & \left(\frac{\mathbf{P_3-P_2}}{w}\right)' \\ 0 & 0 & 0 \end{bmatrix} \tag{4}$$

$$\text{Relative pose transformation matrix } \left(\mathbf{T}_{\{B\}}^{\{C\}}\right) = \begin{bmatrix} \mathbf{R}_{\{B\}}^{\{C\}} & \mathbf{D}_{\{B\}}^{\{C\}} \end{bmatrix} \tag{5}$$

We hypothesize that the two coordinate frames have aligned axes. Thus, the matrix (**BBO**) of the virtual bounding box points' coordinates at the origin of frame {C} can be defined as in (6). Then, the transformation matrix is applied to all previous points' coordinates to find the actual initial points' coordinates matrix (**BBA**) representing the actual pose of the bounding box (7). This process guarantees the

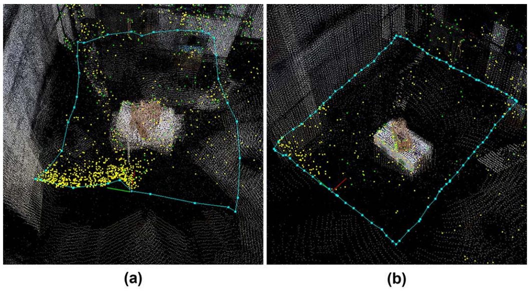


FIGURE 5. Visual SLAM parameter tuning: (a) Mapping performance for the mobile robot moving in a rectangle path with RTAB-Map SLAM default parameter settings. (b) Mapping performance after tunning RTAB-Map SLAM parameter.

generation of a uniform cuboid bounding box.

$$\text{Bounding box points}_{at\ origin of\ \{C\}}(\mathbf{BBO}) = \begin{bmatrix} -L/2 & H/2 & -W/2 & 1 \\ -L/2 & -H/2 & -W/2 & 1 \\ L/2 & -H/2 & -W/2 & 1 \\ L/2 & -H/2 & W/2 & 1 \\ -L/2 & -H/2 & W/2 & 1 \\ L/2 & H/2 & -W/2 & 1 \\ L/2 & H/2 & W/2 & 1 \\ -L/2 & H/2 & W/2 & 1 \end{bmatrix}' \quad (6)$$

$$\text{Bounding box points}_{``actual/initial"\ relative\ to\ \{C\}}(\mathbf{BBA}) = \mathbf{T}^{\{C\}}_{\{B\}}.\mathbf{BBO} \quad (7)$$

C. LOCALIZING THE CAMERA MOVEMENT AND ITERATIVE LABELING

After the user has finished the initial labeling task, the mobile robot moves, acquiring new data for the object of interest from multiple viewpoints. Robot movement to perform relabeling requires a tracking method for the camera or robot. For our implementation, we adopted the RTAB-Map SLAM algorithm [42]. Information from the Microsoft Azure Kinect RGB-D camera is fed to the SLAM algorithm. We assumed that the mobile robot equipped with the RGB-D camera accomplished a quality mapping process for the working environment beforehand. Thus, we had a quality-colored point cloud for the environment where the robot would operate. As RTAB-Map SLAM works for various applications and sensory inputs, it is highly recommended to tune the algorithm parameters to fit the best use case [43]. Fig. 5 reflects the impact of good tunning when the mobile robot moved in a rectangle path.

For our system, the origin of the created map is the world frame {O}. The ROS topic "\rtabmap_localization_pose" report the camera-base frame {CB} relative pose to the world frame {O}. The camera-based frame {CB} is attached rigidly to the camera body and has a fixed and known displacement and orientation relative to the color camera frame {C}. Therefore, it is straightforward to calculate the transformation matrix between the color camera frame and the camera-based frame. The change in the pose between two locations can be calculated by finding the pose difference reported by the ROS topic "\rtabmap_localization_pose" for each location. As such, we can alternatively change any spatial point's representation between previously mentioned coordinate frames when needed.

Relabeling or iterative labeling is an autonomous process associated with mobile robot movement. The camera pose difference relative to the world frame is used to update the bounding box's pose. To trigger the iterative labeling process, the user makes the "start scan gesture" after accomplishing the initial labeling. Considering how the camera is mounted over the mobile robot and assuming that the robot is moving over a flat ground indoors, the orientation of the camera body is possible only around the camera frame's Y-axis. Moreover, the displacement in the vertical direction is invisible. Thus, the pose difference is a 3D vector comprising two displacements (Δd_X and Δd_Z) over the camera frame's X-axis and Z-axis, respectively, and one rotation ($\Delta angle$) around the Y-axis, keeping in mind that the camera's initial pose is the one during the initial labeling stage. Therefore, the pose difference is calculated by finding the difference between the current and initial poses whenever the mobile robot moves.

The pose difference transformation matrix ($\mathbf{Pose_{difference}}$) can be calculated as the product of the rotation difference matrix ($\mathbf{R_{difference}}$) and the displacement difference matrix ($\mathbf{D_{difference}}$), as in (8). The ($\mathbf{pose_{difference}}$) transformation matrix is used to find the updated bounding box points' coordinates (**BBU**), referring to (9). Following these steps, we can relabel the object of interest whenever the mobile robot changes its location.

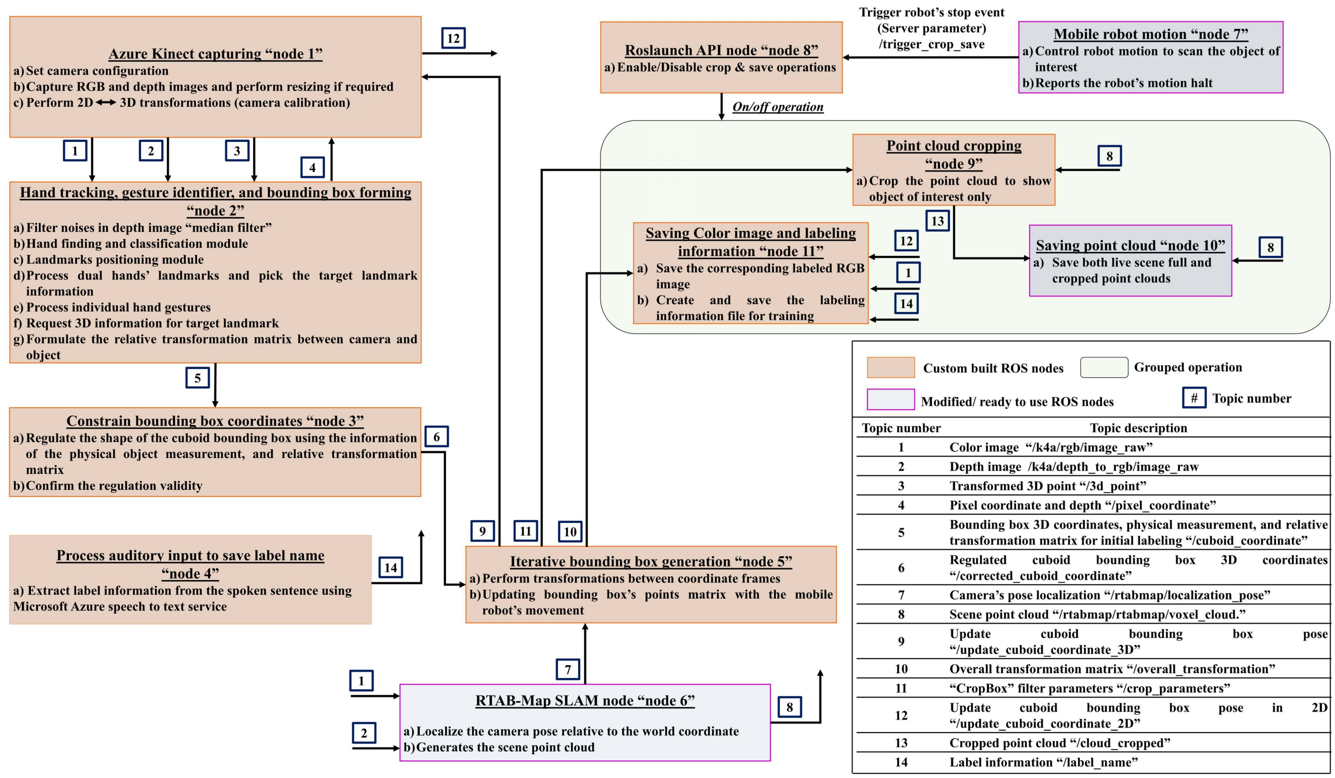
$$\mathbf{Pose_{difference}} = \mathbf{R_{difference}}.\mathbf{D_{difference}} = \begin{bmatrix} \cos(\Delta angle) & 0 & \sin(\Delta angle) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\Delta angle) & 0 & \cos(\Delta angle) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & \Delta d_X \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta d_Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (8)$$

$$\text{Bounding box points}_{\text{update}}(\mathbf{BBU}) = \mathbf{Pose_{difference}}.\mathbf{BBA} \quad (9)$$

If the mobile robot travels over a non-flat surface, it is possible to monitor the six-dimensional change in orientations and displacements. Thus, the displacement in the vertical direction and the rotation around the camera frame's X-axis and Z-axis should be monitored and included in the calculations.

D. DATASET AUTOMATIC CONSTRUCTION

A notable feature of our labeling framework is its ability to construct labeled colored image and point cloud datasets simultaneously. We designed multidata types storing procedure utilizing the bounding box 3D information availability


Azure Kinect capturing "node 1"
a) Set camera configuration
b) Capture RGB and depth images and perform resizing if required
c) Perform 2D ⟷ 3D transformations (camera calibration)
Hand tracking, gesture identifier, and bounding box forming "node 2"
a) Filter noises in depth image "median filter"
b) Hand finding and classification module
c) Landmarks positioning module
d) Process dual hands' landmarks and pick the target landmark information
e) Process individual hand gestures
f) Request 3D information for target landmark
g) Formulate the relative transformation matrix between camera and object
Constrain bounding box coordinates "node 3"
a) Regulate the shape of the cuboid bounding box using the information of the physical object measurement, and relative transformation matrix
b) Confirm the regulation validity
Process auditory input to save label name "node 4"
a) Extract label information from the spoken sentence using Microsoft Azure speech to text service
Iterative bounding box generation "node 5"
a) Perform transformations between coordinate frames
b) Updating bounding box's points matrix with the mobile robot's movement
RTAB-Map SLAM node "node 6"
a) Localize the camera pose relative to the world coordinate
b) Generates the scene point cloud
Roslaunch API node "node 8"
a) Enable/Disable crop & save operations
Trigger robot's stop event
(Server parameter)
/trigger_crop_save
Mobile robot motion "node 7"
a) Control robot motion to scan the object of interest
b) Reports the robot's motion halt
On/off operation
Point cloud cropping "node 9"
a) Crop the point cloud to show object of interest only
Saving Color image and labeling information "node 11"
a) Save the corresponding labeled RGB image
b) Create and save the labeling information file for training
Saving point cloud "node 10"
a) Save both live scene full and cropped point clouds
1
2
3
4
5
6
7
8
9
10
11
12
13
14
Custom built ROS nodes
Grouped operation
Modified/ ready to use ROS nodes
# Topic number

Topic number	Topic description
1	Color image "/k4a/rgb/image_raw"
2	Depth image /k4a/depth_to_rgb/image_raw
3	Transformed 3D point "/3d_point"
4	Pixel coordinate and depth "/pixel_coordinate"
5	Bounding box 3D coordinates, physical measurement, and relative transformation matrix for initial labeling "/cuboid_coordinate"
6	Regulated cuboid bounding box 3D coordinates "/corrected_cuboid_coordinate"
7	Camera's pose localization "/rtabmap/localization_pose"
8	Scene point cloud "/rtabmap/rtabmap/voxel_cloud."
9	Update cuboid bounding box pose "/update_cuboid_coordinate_3D"
10	Overall transformation matrix "/overall_transformation"
11	"CropBox" filter parameters "/crop_parameters"
12	Update cuboid bounding box pose in 2D "/update_cuboid_coordinate_2D"
13	Cropped point cloud "/cloud_cropped"
14	Label information "/label_name"

FIGURE 6. Custom-built ROS node design tree to register 3D feature points, render the initial bounding box's pose, update the bounding box's pose, and save the object dataset automatically.

throughout the labeling process. As described before, we input the feature points and render both the initial and updated bounding box in the 3D domain. Our system uses the depth maps and color images provided by the Azure Kinect camera to create point clouds using the RTAB-Map SLAM algorithm. Thus, we can allocate the rendered bounding box in a native 3D environment like the point cloud. During the process, we monitored the live feed of the point cloud message by subscribing to the relative ROS topic. Whenever the mobile robot reports a motion halt, we work to save the point cloud scene with the corresponding updated coordinates of the bounding box for labeling.

The point cloud saving is associated with the stop event to ensure that the lag in the SLAM's tracking algorithm does not affect the labeling quality. We can save an entire point cloud scene or only the cropped point cloud area containing the object of interest. To set the cropped point cloud volume, we used the "CropBox" filter provided by the point cloud library [44]. The "CropBox" filter was used in a ROS node receiving fixed and variable parameters. The fixed parameters are the object's length (L), height (H), and width (W), whereas the variable parameter is the bounding box updated pose. A roslaunch API node was designed to regulate the cropping and saving of the labeled point clouds and color images, using the robot's motion halt as a trigger event. We transformed the updated bounding box's 3D coordinates to its corresponding 2D pixel coordinates to label the color images with a 2D bounding box.

At each saving event, we stored extra information such as the relative pose of the object to the world frame, the 3D and 2D coordinates of the labeling bounding box, and the approximate physical dimension of the object. We saved this information along with its digital correspondent labeled color images in.''png'' format and labeled point clouds in.''ply'' format. Constructing our datasets as such allows their adoption for training various deep learning models. For example, if we need to train the Yolo model family [45], minor postprocessing steps are required to export the required annotation format, keeping in mind that the collected datasets entirely comprise real-world data that well-depict the appearance change of each included class in 2D and 3D.

The design of ROS nodes for automatic saving and the entire design tree for several predescribed operations inside our labeling framework is shown in Fig. 6.

IV. EXPERIMENT AND SYSTEM EVALUATION

In this study, we propose a labeling framework to solve practical object detection problems encountered by mobile robots. Our framework involves humans for the initial stage of the labeling loop, whereas mobile robots handle the rest of the operation. It was essential to assess each framework component because they all contribute to the system's performance and usability. We started by evaluating the performance of the hand gesture identifier as the essential channel to communicate with the mobile robot. Afterward, we assess the 3D pointing accuracy, showing its influence to

TABLE 1. Hand gesture identifier's performance.

Gesture	Number of attempts	Wrong identifications	Accuracy
Pointing	309	0	100%
Canceling	63	0	100%
Start scanning	41	0	100%
Confirmation	413	0	100%

determine the initial bounding box accuracy and the object's approximate physical dimensions. Next, we test the accuracy of camera tracking, which allows the iterative labeling task and multipurpose training data collection. Finally, we checked our labeling framework's different usability aspects against a publicly available labeling tool by inviting human participants.

A. ACCURACY OF HAND GESTURE IDENTIFIER AND INPUT OF LABEL NAME

A user of our labeling framework can input the labeling data through visual and auditory channels. Visual input controls registering or canceling of feature points and triggers the event to collect data autonomously. To input the visual information, we communicate with the mobile robot using a set of hand gestures. In this section, we first assess hand gesture performance. Then, we show how a user can pass several auditory information layers (e.g., object name, to whom this object belongs, and object location) in the label name.

We observed the hand gesture identifier's performance while placing 41 labels. Two people with different hand sizes and skin tones collaborated to accomplish this task. We passed the six feature points for each label using a pointing gesture, where each input gesture was combined with a confirmation gesture (left hand's closed palm). In a few labeling trials, we tried to cancel some registered feature points using the canceling gesture, which should also be combined with the confirmation gesture. After a complete labeling trial, we tested the ''start scanning'' gesture once. Table 1 shows the hand gesture identifier's performance by counting the misclassified gesture attempts.

Table 1 indicates the superior performance in discriminating between different hand gestures. Our application demands a small set of hand gestures that may be expanded if needed. So far, the hand gesture identifier is an efficient technique to pass commands to the mobile robot considering the application demand.

Aside from evaluating the hand gesture, we experienced a loss of hand tracking sometimes during the operation. Two minor precautions a user should follow to address this issue are as follows: 1) the distance between the user and camera should not exceed 2 m, as we set the Azure Kinect camera to work in the narrow field-of-view unbinned mode for the depth camera and with 1280 × 720 resolution for the color camera [46]; 2) the user should keep his hands in clear view facing the camera so tracking can be resumed.

TABLE 2. Examples of label names recognized by Microsoft Azure speech-to-text service.

Descriptive label name	How a user speaks the label name?
Portable scanner	"start" "**Portable scanner**" "end"
Charging hub used by Alex and usually placed in the living room	"start" "**charging hub**" "space" "**Alex**" "space" "**living room**" "end"

TABLE 3. 3D positioning accuracy.

Feature points coordinates	X-axis average error in cm	Y-axis average error in cm	Z-axis average error in cm
Raw coordinates	2.73	5.52	1.80
Corrected coordinates	**1.13**	**1.44**	**1.37**

Moving to the auditory input, a user can stand in front of the mobile robot and register the label name. To display how we designed to register the label name, Table 2 lists some examples. Our purpose is not to evaluate the Microsoft Azure speech-to-text service but to demonstrate how we can use this robust tool to register a detailed descriptive label name. That brings a new extent to the mobile robot's awareness when discovering its surroundings.

B. 3D POINTING PERFORMANCE

The bounding box's placement depends on how precise the feature point positions are. When a user intends to register a spatial point as a feature point, the index fingertip should be placed over that point. Thus, we assessed the accuracy of positioning the index fingertip by finding the relative positioning ground truth values.

We tested our framework's labeling performance against eight objects of diverse sizes and colors by positioning them at various distances from the camera (Fig. 7). For the first object in Fig. 7(a), the ground truth positions of the first four feature points were recorded relative to the color camera frame. While assigning each feature point, a direct calculated ''raw'' 3D representation of the user's index fingertip position was saved for comparison with the ground truth values. In addition, each feature point's correlated coordinates on the rendered cuboid bounding box were saved for the same purpose. We repeated this process five times to check for performance consistency. Theoretically, there should be no difference between the direct assigned 3D coordinates and their correlated values in the rendered bounding box. However, this is not the case, as we reprocessed all feature points and examined them for possible faulty values to generate the most unified and sensible 3D bounding box. Table 3 displays the two cases' average error values for each coordinate axis.

We selected the first object in Fig. 7(a) for this test as it was located far from the camera, and the height values of the assigned feature points showed the lowest consistency. However, after the reprocessing for all assigned feature points,

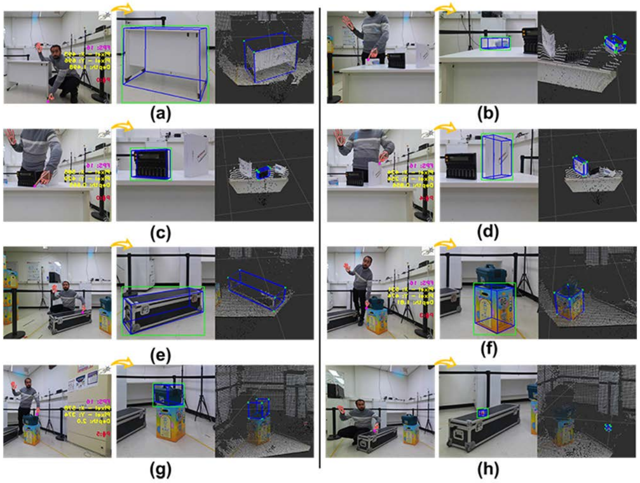


FIGURE 7. Positioning accuracy and initial labeling for different objects in various situations: (a) "Object 1: Study table," labeling relatively large objects placed far from the mobile robot, (b, c, and d) "Object 2: Humidifier, Object 3: Battery charger, and Object 4: Sensors set," labeling small objects with different color tones placed near the mobile robot, (e) "Object 5: Storage box," labeling short-height and dark-colored objects placed far from the mobile robot, (f) "Object 6: Helium gas container," labeling far placed object entangled with another object, (g) "Object 7: Toolbox," labeling a non-fully cuboid object entangled with another object, and (h) "Object 8: Tea pack," labeling tiny and far placed object.

the average pointing accuracy became less than 1.5 cm for all axes. The achieved pointing accuracy allows the user to label any object with trust for the labeling outcome. In Fig. 7, we display the rendered bounding box in the color images and point clouds for eight objects. The tight fit of bounding boxes for all objects reflects the enhanced positioning accuracy. The eighth object in Fig. 7(h) represents a relatively small object (5.8 × 5.6 × 7.2 cm^3) placed far from the camera (almost 2 m away). It is usually difficult to place an accurate 3D label in this case. However, our labeling framework makes it possible and precise. We tested the same for dark-body objects, entangled objects, and objects that are not entirely modeled as cuboids. For all cases, our labeling framework facilitated the allocation of a high-quality bounding box that did not require further modifications or correction. A demonstration video of labeling the same objects in Fig. 7 and other geometry objects "non-cuboid objects" can be watched [47].

Finally, to confirm the accurate 3D pointing of our framework, we assessed the generated object's dimensions by comparing them with the object's actual dimensions. Fig. 8 shows the box plot of the dimension errors for the eight objects.

From Fig. 8, the maximum dimension error does not exceed 2 cm. The highest dimension error appears for the objects' width and is recorded for "Object 1: Study table." When labeling "Object 1," we passed the feature point $\mathbf{P}_3$ as the most distant point from the camera with a depth value of 2.4 m. Considering the degradation of depth accuracy for the far registered point, we can understand the reason for registering the highest width error for "Object 1."

For most length and width values, the generated dimensions were greater than the actual ones, indicating that the actual length and width of the object of interest can be completely contained within the rendered bounding box. However, the height values for the rendered bounding boxes were marginally less than the actual objects' heights in most cases. When assigning feature points using our fingertip, we often have difficulty locating the lower feature point. This human limitation adds a fractional error to the height values. In general, the calculated dimensions well-represent the actual objects' dimensions. This analysis helped us understand how the initial labeling functions vary in diverse situations and will lead to some enhancements, as we illustrate in the next section.

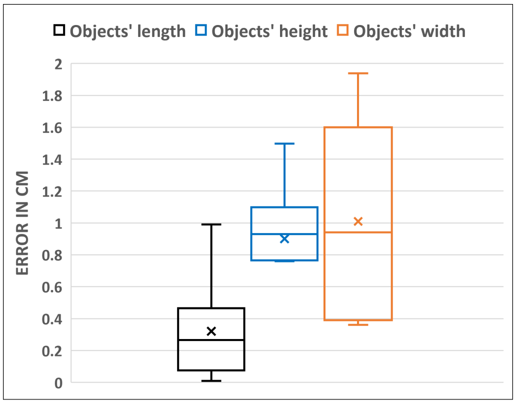


FIGURE 8. Box plot of dimensions' error.

C. CAMERA TRACKING AND LABELING FRAMEWORK OUTPUT

Tracking camera movement facilitates the relabeling operation when the mobile robot moves around. We track the variation in displacement and rotation to update the bounding box pose whenever the mobile robot moves. To track the RGB-D camera and the mobile robot pose, we used visual SLAM (RTAP-Map SLAM). We designed an evaluation procedure to test the tracking performance accuracy with visual SLAM. Proper tuning for mapping and tracking parameters combined with precise mapping for the working environment is required before testing the tracking performance.

We performed this experiment in a workspace of 2.2 × 2.7 m^2 (Fig. 9). Mapping this space starts from a known location representing the origin of the world coordinate $\{\mathrm{O}\}$. The locations where the mobile robot should stop were preplanned to evaluate the tracking performance of visual SLAM (i.e., 1st location, 2nd location, up to 8th location). The ground truth variation values of linear displacements (Δd_X and Δd_Z) and rotation ($\Delta angle$) are known for all locations.

We positioned an object to be labeled inside the workspace. Then, feature points were assigned to perform the initial labeling. At this instant, the location where the robot was standing was considered the first location where initial labeling information should be saved. The "start scanning" gesture was then made to trigger the mobile robot's movement. Afterward, the robot was positioned precisely

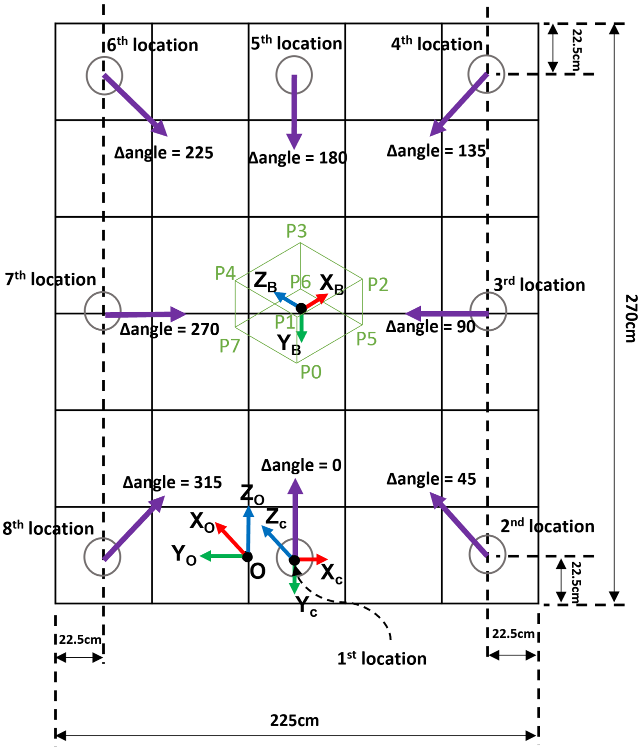


FIGURE 9. Schematic of the workspace and the planned locations for the mobile robot's movement.

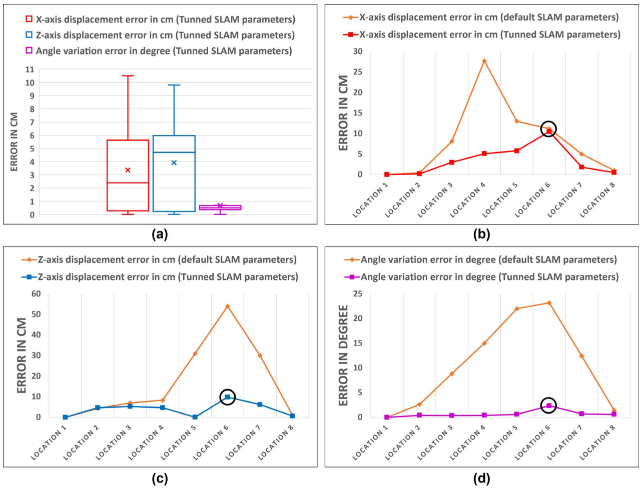


FIGURE 10. Assessing visual SLAM (RTAP-Map SLAM) tracking accuracy and the effect of tunning SLAM's parameters: (a) box plot to show SLAM localization accuracy, (b) effect of tunning SLAM's parameters for the tracking robot linear displacement over X-axis, (c) effect of tunning SLAM's parameters for the tracking robot linear displacement over Z-axis, and (d) effect of tunning SLAM's parameters for the tracking robot angle variation around Y-axis.

for each upcoming location to ensure the correctness of the ground truth values of Δdx, Δdz, and $\Delta angle$. The actual values of linear displacements and rotation were stored by subscribing to the ROS topic "\rtabmap_localization_pose." This entire process was repeated for five iterations, and the localization results were almost identical for each iteration.

Fig.10(a) displays the box plot for visual SLAM localization accuracy by measuring the error value between the ground truth displacements and angle variations and the actual ones. The effect on localization accuracy when using the default SLAM setting and tunned setting is shown in Fig.10(b)–(d).

With tunned SLAM's parameters (Fig. 10(a)), the average displacement error over the X-axis and Z-axis were **3.36** and **3.93** cm, respectively, whereas the average error for the angle variation around the Y-axis was **0.68**°. The localization accuracy was the lowest at the 6th location; we marked the error values with a black circle for this location in Fig. 10(b)–(d). Except for this case, the tracking for rotation angle was excellent for all other locations and recorded values less than 1°. Therefore, it was possible to perform relabeling following (9). Employing these outcomes, we set a specific solution to treat this few-centimeter inaccuracy for the linear displacements by adding offset values to the length and width of the object's bounding box. Similarly, a minimal offset could be added to the height direction to compensate for a human limitation when assigning corners or hard-to-reach points.

The iterative labeling process for color images and point clouds with active visual SLAM tracking running in the background is shown in Fig. 11. The resultant 2D bounding box "green color" properly contains the object of interest in the color image. In addition, the 3D bounding box "blue color" fully contains the point cloud segment of the object of interest.

Finally, each stop for the mobile robot is combined with a saving process for the labeled color image, labeled point cloud, and label-related information file. The label-related information file includes the object's relative pose to the world frame, the object's physical dimensions, and the coordinates of the bounding boxes for the color image and point cloud. The media file in [47] shows the iterative labeling process and the dataset's automatic saving for several objects.

D. USER STUDY

1) DEMOGRAPHICS AND PROCEDURE

We invited six participants (four males and two females) to perform our experiment. The participants' age ranged from 19 to 34 years old (Mean = 29.17; SD = 4.81). They were graduate and postgraduate students with diverse technical backgrounds (engineering, biomedical, physics, etc.), and all of them had experience using PC and working with different software. None of them have tried labeling data for training before. The participants did not report any mobility or vision problems (e.g., problems using their hands or watching).

As a preparation step for this test, eight colored images and a similar number of correspondent point cloud scenes were gathered beforehand (Fig. 12). We tried to reflect an actual scene where some laboratory equipment stands side by side. In the shown sequence of images and point

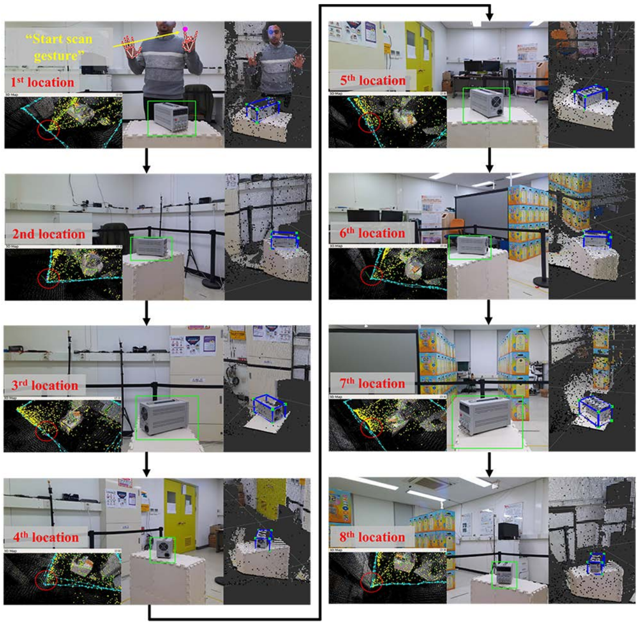


FIGURE 11. Iterative labeling for color images and point clouds.

clouds, we were interested in labeling the middle object "the power supply." The collected data were distributed to all participants to label the object using two freely available annotation software for labeling colored images [23] and point clouds [48]. We chose these two software as representatives for their respective categories because they both have a simple user interface and are easy to learn. The participants were also asked to label that object using our labeling framework (Fig. 13). This part aims to test the difference between the two methods, considering various human factors (e.g., task accomplishment time, usability evaluation, workload assessment, and general usage observation and comments).

We designed a counterbalanced within-group experiment to reduce the learning effect by making participants try the two methods and shuffling the starting order. Half of the participants started using our labeling framework, whereas the other half started using the labeling software. In the transition between the two methods, all participants were asked to take a break for 10 min.

2) METHOD 1: LABELING USING FREELY AVAILABLE SOFTWARE

We explained the experiment procedure and briefly described the software functionality that participants would use before allowing them to listen to extra explanations using available YouTube videos for each software. Then, we allowed all participants to experience each software for approximately 10 min (e.g., practicing labeling the official example point cloud provided by [48]). That procedure starts with delivering the previously collected data shown in Fig. 12, so each participant can use the labeling software to place 2D and 3D bounding boxes properly.

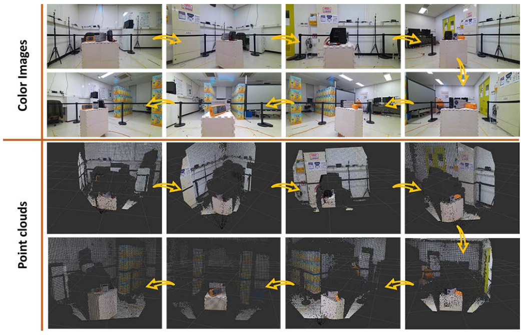


FIGURE 12. Data delivered to participant for labeling using software.

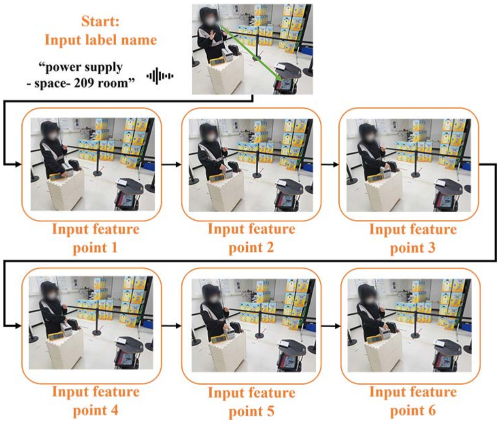


FIGURE 13. Participants passing labeling names and feature points using our labeling framework.

3) METHOD 2: LABELING USING OUR FRAMEWORK

In the beginning, all participants were introduced to the hardware used for our labeling framework. For the next 10 min, we instructed them about the technique to pass feature points and how to register the label name. Then, we asked each participant to speak the label name and place the six feature points (Fig. 13). The participants observed the created initial 2D and 3D labels throughout the screen installed over the mobile robot. Afterward, the participants watched the remaining automatic procedure to relabel and save for the "power supply" dataset.

4) QUALITATIVE AND QUANTITATIVE ANALYSES

At the start of the experiment, all participants provided their personal information by answering preliminary questions through a predesigned survey. While participants were trying each method, we noted any difficulties using the labeling method, wrote down participants' verbal comments, and recorded the time taken by each participant to accomplish the labeling task. After the participants tried the two labeling methods, they were asked to fill out a system usability scale (SUS) [49] to evaluate their usability experience separately for each method. Then, they continued filling out the National Aeronautics and Space Administration Task Load Index (NASA-TLX) [50] to measure the perceived workload relative to each method.

Labeling difficulties and participants' verbal comments, all participants were observed in the learning phase and while doing labeling tasks. In the learning phase of using the first method (freely available software), there were no comments from the participants. Few participants watched the learning videos multiple times to adapt to the software tools. When the participants started labeling the provided data shown in Fig. 12, they decently performed the color image labeling. However, two participants did not care about the label's quality but focused more on performing fast. The missing label of one image was repeated by two other participants.

The story was different when the participants started labeling the point cloud data; almost all participants found it challenging to place the 3D bound box. None of them performed this task completely. It was difficult to label while manipulating the point cloud and avoiding occlusion with other objects surrounding the object of interest. Although the point cloud labeling software provides the labeling tool intuitively, some participants asked to watch the learning videos again, as they thought they had missed some information. Three participants avoided labeling the object directly; instead, they created a dummy general cuboid shape bounding box and manipulated it to fit the dimensions of the object of interest. Some participants commented, "After I labeled two point clouds, I feel too bored to continue the process" and "Keep manipulating the point cloud to fit the bounding box makes me feel tired."

The participants reported no issues in the learning phase for the second method (our labeling framework). However, in the practicing phase, they were a bit confused about the duration of closing the left hand palm to make the confirmation gesture. All of them adapted to performing the confirmation gesture after a few minutes of learning. This observation was important for us as it guided us to use some visual indicator for showing the counting down when using the confirmation gesture. In the primary process of labeling the object of interest, all participants had no trouble entering the six feature points. Only one participant experienced non-solid fitting of the bounding box, as he pointed to the wrong locations on the object of interest. Overall, the success rate for our method to precisely label the object of interest in color images and point clouds is no comparison with the labeling software, particularly when working with point clouds. All participants reported satisfaction watching the resultant automatic and simultaneous labeled color images and point clouds during the iterative labeling phase. One participant commented, "Using hand gestures to label an object was fun and easy; I have to memorize the hand gesture, but once I do, labeling objects is not an effort anymore."

Labeling task accomplishment time, we measured the time taken by all participants from the time they started the main labeling task to the time they decided to stop. When using the first method, the average times taken by all participants to label the color images and point clouds were **4** and **27** min, respectively. Although they took a long time to label the point clouds, the label quality was sometimes inaccurate or wrong. These times are the average time to label only a set of eight color images and point clouds. The time taken will increase if the amount of data to be labeled is larger.

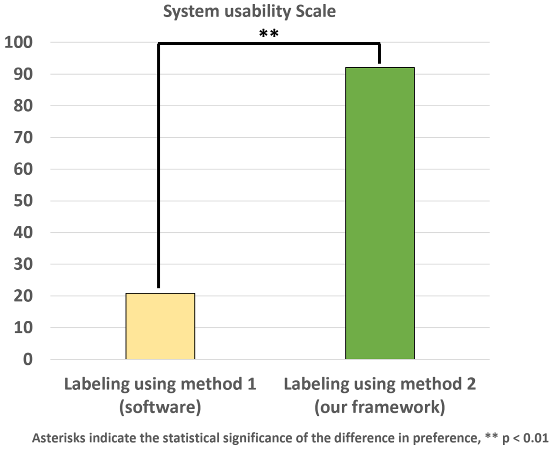


FIGURE 14. Average SUS scores.

Moreover, when calculating the average time to accomplish the labeling task using our labeling framework, we found it to be **36**s,keeping in mind that most color images and point clouds have accurate labels. In this case, the time to label eight color images and point clouds is the same, even if it was demanded to label a larger data size. The user does not have to invest more time in labeling, as the automated agent will accomplish the remaining job (e.g., the mobile robot in our case).

The results demonstrated above can be observed throughout the outcomes of standard qualitative tests such as SUS and NASA-TLX.

Usability test: subjective usability was compared and identified on how easy and efficient the use of our labeling framework in contrast to the labeling software using SUS. The SUS was evaluated on a scale of 0 to 100 in increments of 10 with five steps anchored with "Strongly Disagree" to "Strongly Agree." SUS percentage scores were described in [49]; scores greater than 71 points indicate that the system is acceptable.

The one-way repeated measure ANOVA test indicates a significant difference (F (1, 11) = 129.500, p <**.000**) in SUS. The score for the labeling framework was higher (*Mean* = **92.08**, *SD* = 7.31) than the score of the labeling software (*Mean* = **20.83**, *SD* = 13.47) (Fig. 14). This result does not mean that the usability of the labeling software is extremely bad itself, but it indicates that the usability of the labeling software was low relative to our labeling framework (Fig. 14).

For the **workload test**, the perceived workload was measured by NASA-TLX, a subjective workload tool to determine an overall workload rating, which measures the level of six factors (mental demand, physical demand, temporal demand, effort, performance, and frustration). Each factor is evaluated on a scale of 0 to 100 [50].

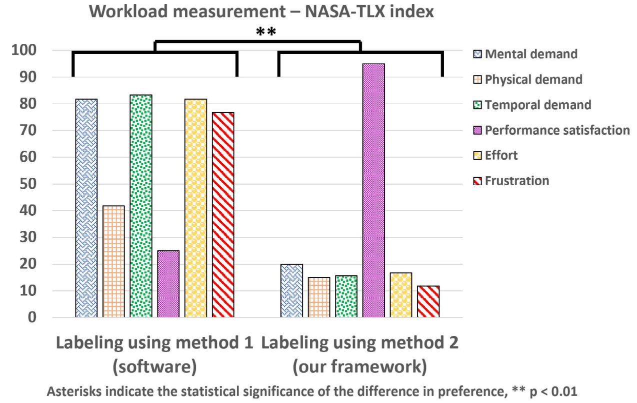


FIGURE 15. NASA-TLX workload scores for trying the two labeling methods.

The ANOVA test shows a significant difference in the overall NASA-TLX workload between our labeling framework and the labeling software (F (1,11) = 48.517, $p < \mathbf{0.000}$). The perceived workload was higher while using the labeling software (*Mean* = **70.83**, *SD* = 19.37) than using our labeling framework (*Mean* = **14.81**, *SD* = 3.57) (Fig. 15).

V. DISCUSSION

Our labeling framework delivers an intuitive interaction between humans and mobile robots. The simple hand gesture was effective to pass the essential labeling information. The hand gesture identifier was assessed quantitatively, showing an excellent performance. Besides, the qualitative assessment reflects the easiness to use this input modality. Combining two hands for passing the gesture, one to perform the main gesture and the other for confirmation, was praised by the participants as it gives them the control not to enter data unintentionally. The participants reported no issues learning this communication method.

We proved a high accuracy for positioning spatial points and performing the initial labeling task. The variety of objects we can label in practical environments increases our confidence in using our system. There was a significant difference for all participants in how easy and intuitive our labeling framework generates the labels compared with conventional methods. Our original plan was to invite more participants; however, the similarities of the collected data in the small sample were convincing enough that increasing the sample would not change the final output.

For the iterative labeling, we observed functioning but not solidly fitted labels such as the one generated from the initial labeling stage in Fig. 7. This behavior results from the loss in tracking performance, which could be enhanced by improving the tracking method. Developing a novel tracking method or using expensive tracking equipment is not the focus of this study. However, our primary goal is to demonstrate the visible and complete labeling framework to assist mobile robots in overcoming the difficulties associated with object detection in a real-world scenario. That framework removes the discrepancy between training data and actual environmental scenes and elevates the burden of using a specialized tool for labeling 2D or complex 3D data. This study demands reducing the time and effort required to generate in-house annotated data. Further, our framework generates multitype-labeled datasets suitable for training models that perform diverse duties. We believe our labeling framework could work as a practical, real-time, and efficient acquiring tool to prepare effortless training datasets employed by mobile robots.

The fact that the entire framework relies on a single sensing device (i.e., the RGB-D camera) simplifies the design. Notably, this sensory input is installed over a mobile robot; thus, a massive upgrade to our framework could be achieved by incorporating other possible robots' sensory inputs.

Instead of relying on pretrained deep learning models, a robot's owner can guide the robot to understand the surroundings in a better way. For example, suppose a robot is operating in an indoor space and fails to recognize some objects. In that case, the robot can be guided to label the objects using our simple interactive technique, generating a dataset for each object, and later training the model with the newly collected data. Another technique to use this framework is to use it as the only channel to collect the training data; this could be a long process to label all objects in the surrounding space. However, this generates a full real-world dataset for objects with their actual dimensions and multiviewpoint appearance. The impact of such labeled data is strong on how robots perceive and interact with their surroundings. The idea of training a model with big data is attractive and successful, but what if the indoor used robots are exposed to a limited set of objects? In this case, ensuring that the training data reflects the actual input scenes for the robots will contribute to solving several problems. Then, state-of-art deep learning models (e.g., 2D and 3D object detection and 3D pose estimators) could be used with mobile robots not only in a controlled environment but also in real-human environments and show high detection and estimation rates.

VI. CONCLUSION AND FUTURE WORK

In this study, we present the full design and detailed assessment of a labeling framework that serves field mobile robots. Our presented framework includes a single sensory input device, basic form of a mobile robot, and custom-designed ROS nodes. The design of our framework guarantees no usage of software or physical tools to label or refine the labeled data from the end user. In addition, it allows the end user to use intuitive communication channels to pass commands to the robot. The assessment of our framework shows strong practical and user acceptance evidence for its effectiveness. Our labeling framework contributes to the mobile robot field by providing an efficient human-robot collaborative method to acquire and label real-world multitype training data on demand. Thus, it helps accelerate the integration and spread of mobile robots into various human service sectors. Indeed, we believe that training mobile robots with real-world data can ingrain a realistic awareness and better interactions with their surroundings.

For future work, we will design more solid camera tracking techniques using sensory data available within the mobile robot platform or, if needed, incorporate a global positioning technique. In addition, we will enlarge our collected multitype datasets by collecting data for diverse objects. The collected datasets will be used to train diverse state-of-art deep learning models to prove their quality. Although we demonstrated the possibility of labeling non-cuboid shape geometries in the demo video [47], we are planning to expand our concept to label any geometry intuitively while maintaining the user hands-free and eliminating the use of any graphical user interface.

ACKNOWLEDGMENT

The authors would like to thank Aya Ataya and Sanghun Jeon, for their help with editing and experiments.

REFERENCES

[1] S. Hoque, M. Y. Arafat, S. Xu, A. Maiti, and Y. Wei, "A comprehensive review on 3D object detection and 6D pose estimation with deep learning," *IEEE Access*, vol. 9, pp. 143746–143770, 2021.

[2] J. Sandino, F. Vanegas, F. Maire, P. Caccetta, C. Sanderson, and F. Gonzalez, "UAV framework for autonomous onboard navigation and people/object detection in cluttered indoor environments," *Remote Sens.*, vol. 12, pp. 1–31, Oct. 2020.

[3] J. Ma, Y. Ushiku, and M. Sagara, "The effect of improving annotation quality on object detection datasets: A preliminary study," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2022, pp. 4850–4859.

[4] L. Wang, R. Li, J. Sun, X. Liu, L. Zhao, H. S. Seah, C. K. Quah, and B. Tandianus, "Multi-view fusion-based 3D object detection for robot indoor scene perception," *Sensors*, vol. 19, no. 19, p. 4092, Sep. 2019.

[5] G. Baruch, Z. Chen, A. Dehghan, T. Dimry, Y. Feigin, P. Fu, T. Gebauer, B. Joffe, D. Kurz, A. Schwartz, and E. Shulman, "ARKitScenes: A diverse real-world dataset for 3D indoor scene understanding using mobile RGB-D data," 2021, *arXiv:2111.08897*.

[6] M. Suchi, T. Patten, D. Fischinger, and M. Vincze, "EasyLabel: A semi-automatic pixel-wise object annotation tool for creating robotic RGB-D datasets," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 6678–6684.

[7] J. Cartucho, R. Ventura, and M. Veloso, "Robust object recognition through symbiotic deep learning in mobile robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 2336–2341, 2018.

[8] A. Munappy, J. Bosch, H. H. Olsson, A. Arpteg, and B. Brinne, "Data management challenges for deep learning," in *Proc. 45th Euromicro Conf. Softw. Eng. Adv. Appl. (SEAA)*, Aug. 2019, pp. 140–147.

[9] J. Brownlee, "Data preparation for machine learning," Tech. Rep., 2022.

[10] P. Wspanialy, J. Brooks, and M. Moussa, "An image labeling tool and agricultural dataset for deep learning," 2020, *arXiv:2004.03351*.

[11] Q. Meng, W. Wang, T. Zhou, J. Shen, L. V. Gool, and D. Dai, "Weakly supervised 3D object detection from lidar point cloud," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 515–531, 2020.

[12] *MediaPipe Hands*. Accessed: Sep. 18, 2022. [Online]. Available: https://google.github.io/mediapipe/solutions/hands.html

[13] S. Hinterstoisser, O. Pauly, H. Heibel, M. Martina, and M. Bokeloh, "An annotation saved is an annotation earned: Using fully synthetic training for object detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshop (ICCVW)*, Oct. 2019, pp. 2787–2796.

[14] A. Dehban, J. Borrego, R. Figueiredo, P. Moreno, A. Bernardino, and J. Santos-Victor, "The impact of domain randomization on object detection: A case study on parametric shapes and synthetic textures," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Nov. 2019, pp. 2593–2600.

[15] B. Kiefer, D. Ott, and A. Zell, "Leveraging synthetic data in object detection on unmanned aerial vehicles," 2021, *arXiv:2112.12252*.

[16] F. E. Nowruzi, P. Kapoor, D. Kolhatkar, F. Al Hassanat, R. Laganiere, and J. Rebut, "How much real data do we actually need: Analyzing object detection performance using synthetic and real data," 2019, *arXiv:1907.07061*.

[17] A. Tsirikoglou, J. Kronander, M. Wrenninge, and J. Unger, "Procedural modeling and physically based rendering for synthetic data generation in automotive applications," 2017, *arXiv:1710.06270*.

[18] M. Wrenninge and J. Unger, "Synscapes: A photorealistic synthetic dataset for street scene parsing," 2018, *arXiv:1810.08705*.

[19] Y. Movshovitz-Attias, T. Kanade, and Y. Sheikh, "How useful is photo-realistic rendering for visual learning," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 202–217.

[20] Z. Zhao and X. Chen, "Building 3D semantic maps for mobile robots using RGB-D camera," *Intell. Service Robot.*, vol. 9, no. 4, pp. 297–309, Oct. 2016.

[21] Y. Qin, T. Mei, Z. Gao, Z. Lin, W. Song, and X. Zhao, "RGB-D SLAM in dynamic environments with multilevel semantic mapping," *J. Intell. Robotic Syst.*, vol. 105, no. 4, pp. 1–18, Aug. 2022.

[22] B. Pande, K. Padamwar, S. Bhattacharya, S. Roshan, and M. Bhamare, "A review of image annotation tools for object detection," in *Proc. Int. Conf. Appl. Artif. Intell. Comput. (ICAAIC)*, May 2022, pp. 976–982.

[23] *Make Sense Annotation Tool*. Accessed: Oct. 5, 2022. [Online]. Available: https://www.makesense.ai/

[24] *Object Detection (COCO-SSD)*. Accessed: Oct. 5, 2022. [Online]. Available: https://github.com/tensorflow/tfjs-models/blob/master/coco-ssd/README.md

[25] M. Ibrahim, N. Akhtar, M. Wise, and A. Mian, "Annotation tool and urban dataset for 3D point cloud semantic segmentation," *IEEE Access*, vol. 9, pp. 35984–35996, 2021.

[26] H. A. Arief, M. Arief, G. Zhang, Z. Liu, M. Bhat, U. G. Indahl, H. Tveite, and D. Zhao, "SAnE: Smart annotation and evaluation tools for point cloud data," *IEEE Access*, vol. 8, pp. 131848–131858, 2020.

[27] E. Barnefske and H. Sternberg, "Evaluating the quality of semantic segmented 3D point clouds," *Remote Sens.*, vol. 14, no. 3, p. 446, Jan. 2022.

[28] D. T. Nguyen, B.-S. Hua, L.-F. Yu, and S.-K. Yeung, "A robust 3D-2D interactive tool for scene segmentation and annotation," 2016, *arXiv:1610.05883*.

[29] Y.-S. Wong, H.-K. Chu, and N. J. Mitra, "SmartAnnotator an interactive tool for annotating indoor RGBD images," *Comput. Graph. Forum*, vol. 34, no. 2, pp. 447–457, 2015.

[30] F. Wirth, J. Quehl, J. Ota, and C. Stiller, "PointAtMe: Efficient 3D point cloud labeling in virtual reality," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2019, pp. 1693–1698.

[31] *BasicIA. Quickly Annotate With Powerful Multi-Sensory Labeling Tools*. Accessed: Nov. 1, 2022. [Online]. Available: https://www.basic.ai/

[32] *Scale. Better Data Leads to More Performant Models*. Accessed: Nov. 1, 2022. [Online]. Available: https://scale.com/

[33] Playment AI Data Solutions. *Creating and Enhancing the World's Data to Enable Better AI Via Human Intelligence*. Accessed: Nov. 1, 2022. [Online]. Available: https://www.telusinternational.com/solutions/ai-data-solutions?INTCMP=ti_playment

[34] D. D. Gregorio, A. Tonioni, G. Palli, and L. D. Stefano, "Semiautomatic labeling for deep learning in robotics," *IEEE Trans. Autom. Sci. Eng.*, vol. 17, no. 2, pp. 611–620, Apr. 2020.

[35] *Turtlebot3 Waffle Pi*. Accessed: Sep. 10, 2022. [Online]. Available: https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/

[36] *Microsoft Azure Kinect Camera*. Accessed: Sep. 12, 2022. [Online]. Available: https://learn.microsoft.com/en-us/azure/Kinect-dk/hardware-specification

[37] A. Elsharkawy, K. Naheem, D. Koo, and M. S. Kim, "A UWB-driven self-actuated projector platform for interactive augmented reality applications," *Appl. Sci.*, vol. 11, no. 6, p. 2871, Mar. 2021.

[38] W. Xiong, L. Wu, F. Alleva, J. Droppo, X. Huang, and A. Stolcke, "The Microsoft 2017 conversational speech recognition system," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2018, pp. 5934–5938.

[39] *Microsoft Azure Speech Service Documentation*. Accessed: Oct. 19, 2022. [Online]. Available: https://learn.microsoft.com/en-us/azure/cognitive-services/Speech-Service/

[40] S. Jeon, A. Elsharkawy, and M. S. Kim, "Lipreading architecture based on multiple convolutional neural networks for sentence-level visual speech recognition," *Sensors*, vol. 22, no. 1, p. 72, Dec. 2021.

[41] S. Jeon and M. S. Kim, "End-to-end lip-reading open cloud-based speech architecture," *Sensors*, vol. 22, no. 8, p. 2938, Apr. 2022.

[42] *Simultaneous Localization and Mapping Algorithm, RTAB-Map SLAM*. Accessed: Oct. 21, 2022. [Online]. Available: http://introlab.github.io/rtabmap/

[43] *RTAB-Map SLAM Parameters Tunning*. Accessed: Oct. 21, 2022. [Online]. Available: https://github.com/introlab/rtabmap/wiki/Change-parameters

[44] *CropBox Filter in Point Cloud Library*. Accessed: Oct. 2, 2022. [Online]. Available: https://pointclouds.org/documentation/classpcl_1_1_crop_box_3_01pcl_1_1_p_c_l_point_cloud2_01_4.html

[45] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 7263–7271.

[46] *Azure Kinect DK Hardware Specifications*. Accessed: Oct. 5, 2022. [Online]. Available: https://learn.microsoft.com/en-us/azure/Kinect-dk/hardware-specification

[47] *Demonstration Video, Human-Robot Labeling Framework to Construct Multitype Real World Datasets*. Accessed: Nov. 10, 2022. [Online]. Available: https://youtu.be/kiD6TQ8tVAk

[48] C. Sager, P. Zschech, and N. Kuhl, "LabelCloud: A lightweight labeling tool for domain-agnostic 3D object detection in point clouds," *Comput.-Aided Des. Appl.*, vol. 19, no. 6, pp. 1191–1206, Mar. 2022.

[49] A. Bangor, P. Kortum, and J. Miller, "Determining what individual SUS scores mean: Adding an adjective rating scale," *J. Usability Stud.*, vol. 4, no. 3, pp. 114–123, 2009.

[50] G. S. Hart and E. LowellStaveland, "Development of NASA-TLX (task load index): Results of empirical and theoretical research," *Adv. Psychol.*, vol. 52, pp. 139–183, Apr. 1998.

AHMED ELSHARKAWY received the B.S. degree in electronic engineering from Menoufia University, Egypt, in 2012, and the M.S. degree in mechatronics engineering from the Gwangju Institute of Science and Technology, South Korea, in 2017, where he is currently pursuing the Ph.D. degree. His current research interests include ground mobile and flying robots navigation and control, human–computer interaction, haptics, UWB-based indoor localization systems, and artificial intelligence.

MUN SANG KIM (Member, IEEE) received the B.S. and M.S. degrees in mechanical engineering from Seoul National University, Seoul, South Korea, in 1980 and 1982, respectively, and the Dr.-Ing. degree in robotics from the Technical University of Berlin, Berlin, Germany, in 1987. From 1987 to 2016, he was a Research Scientist at the Korea Institute of Science and Technology, Seoul. He led the Advanced Robotics Research Center, in 2000, where he became the Director of the "Intelligent Robot—The Frontier 21 Program," in October 2003, which is one of the most challenging research programs in South Korea. He is currently a Professor with the School of Integrated Technology, Gwangju Institute of Science and Technology. His current research interests include healthcare robotics, UWB-based indoor localization systems, and culture technology.