Received November 16, 2019, accepted December 3, 2019, date of publication December 6, 2019, date of current version January 3, 2020.

Digital Object Identifier 10.1109/ACCESS.2019.2958092

Automatic Elevator Button Localization Using a Combined Detecting and Tracking Framework for Multi-Story Navigation

SHENLU JIANG[1,2], WEI YAO[2,3], MAN-SING WONG[2,3], MENG HANG[4], ZHONGHUA HONG[5], EUN-JIN KIM[1], SUNG-HYEON JOO[1], AND TAE-YONG KUC[1]
[1]College of Information and Communication Engineering, Sungkyunkwan University, Suwon 440-746, South Korea
[2]Department of Land Surveying and Geo-Informatics, The Hong Kong Polytechnic University, Hong Kong
[3]Research Institute for Sustainable Urban Development, The Hong Kong Polytechnic University, Hong Kong
[4]School of Mechanical Engineering and Automation, Beihang University, Beijing 100083, China
[5]College of Information Technology, Shanghai Ocean University, Shanghai 201306, China

Corresponding author: Tae-Yong Kuc (tykuc@skku.edu)

This work was supported in part by the Korea Evaluation Institute of Industrial Technology (KEIT) funded by the Ministry of Trade, Industry & Energy (MOTIE) under Grant 1415162366 and Grant 1415162820, in part by the grant PolyU 1-BBWD from the Research Institute for Sustainable Urban Development, The Hong Kong Polytechnic University, Shanghai Rising-Star Program, under Project 17QB1403800, and in part by the National Natural Science Foundation of China under Project 41871325.

ABSTRACT Simultaneous localization and mapping (SLAM) is an important function for service robots to self-navigate modernized buildings. However, only a few existing applications allow them to automatically move between stories through elevator. Some approaches have accomplished with the aid of hardware; however, this study shows that computer vision can be a promising alternative for button localization. In this paper, we proposed a real-time multi-story SLAM system which overcomes the problem of detecting elevator buttons using a localization framework that combines tracking and detecting approaches. A two-stage deep neural network initially locates the original positions of the target buttons, and a part-based tracker follows the target buttons in real-time. A positive-negative classifier and deep learning neural network (particular for button shape detection) modify the tracker's output in every frame. To allow the robot to self-navigate, a 2D grid mapping approach was used for the localization and mapping. Then, when the robot navigates a floor, the A$*$ algorithm generates the shortest path. In the experiment, two dynamic scenes (which include common elevator button localization challenges) were used to evaluate the efficiency of our approach, and compared it with other state-of-the-art methods. Our approach was also tested on a prototype robot system to assesses how well it can navigate a multi-story building. The results show that our method could overcome the common background challenges that occur inside an elevator, and in doing so, it enables the mobile robot to autonomously navigate a multi-story building.

INDEX TERMS Elevator button localization, multi-story navigation, object detection, visual tracking, deep learning.

I. INTRODUCTION

For a mobile robot, moving between the stories of a modern building is challenging. This capability is required in tasks such as the automatic delivery of goods or automatic transport of patients in hospitals. Autonomous navigation includes two critical tasks: navigation on a single story, and navigation between stories. Many existing approaches which use simultaneous localization and mapping (SLAM) will be reviewed in Section 2. A deficiency in most existing approaches is that the autonomous indoor navigation of service robots only focuses on single-story navigation. For multi-story navigation, robots must use elevators, and most state-of-the-art approaches depend on either human assistance or the collaborative efforts of multiple robots (for example, Miura *et al.* [1] and Veloso *et al.* [30]). In these methods, a human must help the robot operate the elevator; this requirement limits the service robot's autonomous navigation capability. Therefore, a robust method for automatic recognition of the elevator buttons is essential for

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

multi-story SLAM. Because of the wide view angle and detailed information collected, computer vision can solve this problem. However, classical methods such as template matching [9] are sensitive to background changes, and deep learning methods based on convolutional neural networks (CNNs) [11] have high computational costs. Otherwise, the real-time processing is essential for the robot system. If the vision algorithm costs exhaustive computing cost, the entire system, i.e., controlling system and navigation system will get hung because of the large memory and CPU usage through the deep learning procedure leading the entire system to fail.

In this paper, we propose a combined detection and tracking strategy for rapid and robust localization of the buttons. The key idea is to use generative adversarial networks (GANs): combining online and offline detectors. Our approach can be divided into three parts. First of all, we use the detection method based on a two-stage neural network to provide a single-shot detection of the target buttons on the scene. However, detection based on deep learning generally has high computational costs. As elevator buttons have visible edges, we use edge detection (a traditional image processing method) to search for the rectangles, and then input these rectangles into the neural network. We use the visual tracking technology combined with a deep learning neural network because this locates the object with low computational costs, and is adaptive to the environment. We then use our own designed tracking approach to localize the target buttons after they are detected. A pairwise consensus of cluster trackers tracks the elevator through the center of pairwise feature points. In the second step, we propose two-step validation. A positive-negative (PN) classifier checks the tracker's output's accuracy. Then, a deep neural network detects the buttons in the elevator in the appended region of the tracker and PN classifier (it only detects the buttons' shape, not the numbers on them) and modifies the output. Finally, the three outputs are merged as a final region of interest (ROI), and the tracker and PN classifier are updated according to the output. The tracker and PN classifier serve as online detectors to adapt the localization framework to the environment. The deep neural network serves as an offline detector that restricts the object's properties. With this combination, the online tracker's focus is restricted to the target object, and the offline detector is more adaptive to the environment. Thus, our approach is both robust and fast on a mobile platform. Unlike the methods based solely on object detection, the visual tracking allows a smooth and continuous localization of the target with greatly lower computing cost. The methods based on object detection might face the problem of miss-detection and ROI flash, but the tracking will not lose because it simplifies the localization role to only detect the targets. Moreover, the continuous localization can also benefit the controlling module on the robot arm and navigating when the robot approaches the panel. Accumulating errors are inevitable, but the continuous localization assists on correcting those errors, benefiting navigation and arm control tasks. Therefore, the combined strategy is essential for operating the system both with high accuracy and quick response.

Autonomous navigation on each story requires the real-time position of the robot and a map of the environment. As lidar can obtain dense and precise depth information in a plane, our approach uses lidar data to perform two-dimensional laser SLAM. This can be run on a mobile platform in real time. Three tasks should be considered in the navigation module: 1) finding the nearest elevator and relocating to another floor; 2) providing accurate SLAM on each floor; and 3) recalculating the robot's position on the map of the new floor when the robot uses the elevator.

The experiments show that our approach is robust for the task of elevator button localization. Two dynamic image sequences recorded in the elevator are designed to reflect the challenges of this environment (such as blurriness, darkness, light, different angles, or reflections). The results show that our detection strategy always detects the correct target button, even when image quality is poor. Furthermore, the tracking model can constantly localize the target button from different distances and angles, while processing at a high frames-per-second (FPS). In experiments with a real robot, our robot moves between two stories. The results illustrate that thanks to our approach, the robot can complete tasks in a multi-story environment. Throughout its service, the robot autonomously navigates each floor, and localizes its initial position on the map after relocating to a different story. The robust elevator localization also enables our robot to operate the elevator.

The contributions of this paper can be summarized as follows: (1) We demonstrated the efficiency of combined detection and tracking strategies for elevator button localization. (2) We applied our button localization framework on a multi-story service robot, which operating on a central processing unit (CPU) mobile platform.

II. RELATED WORK

A. AUTONOMOUS NAVIGATION

Research on SLAM technology has a long history (a survey can be found in [2]). Several approaches using a camera [33] or lidar have been proposed for this task. Vision SLAM methods with RGB or gray cameras consume significantly more CPU resources than laser SLAM. They are also less accurate when environmental brightness changes. As a result, laser SLAM is more popular in the industry than vision SLAM. In [3], Murphy and Doucet proposed a particle filter called Rao-Blackwellized to solve the SLAM problem. This filter can estimate the robot's position more accurately than ordinary particle filters. However, to obtain a precise localization and mapping algorithm, a large quantity of particles needs to be maintained. This leads to a high time complexity. Grisetti *et al.* [4] proposed a grid mapping algorithm to solve the problems of high complexity and particle degeneration in Rao-Blackwellized particle filters. Luo and Qin [32] presented a fast flier to minimize the computing cost. Using the robot's movement data and 2D depth data from recent frames, a more accurate particle distribution is calculated. This reduces the uncertainty of robot localization in the filter prediction stage; thus slashing the number of particles. Another well-known algorithm is the Hector mapping algorithm [5] which performs as a stable scan matching method. While processing scan matching, the algorithm uses the Gauss-Newton method to obtain the best current position estimation in a fixed number of iterations. This method also matches the versions of the established map in different resolutions, thus avoiding convergence to the local minima. Google released its Carto Graher 2D SLAM technology in 2016 [6]. In addition to the front-end matching calculation, it performs the branch-and-bound graph optimization in the backend. The established map is divided into multiple subgraphs. Each subgraph has a corresponding key frame. During loop closure detection, the robot matches the current laser scan with the subgraph keyframes to split the accumulated errors into their appropriate subgraphs. However, mapping the robot's movement between stories is a challenge, as it is difficult for it to control the elevator and plan its path. The current trend is to utilize human assistance [1] or employ collaborative robot systems [30]. However, these methods demand extra resources to allow the robot to utilize the elevator. Therefore, a fully automatized method is needed.

B. BUTTON DETECTION AND RECOGNITION

To help the robot operate the elevator autonomously, we processed image data to obtain the positions of the elevator buttons on the scene. Related research can be divided into two areas: button detection and button recognition.

In button detection, some methods use visual features of elevator button, such as color or texture. Yu *et al.* [7] presented a system to detect elevator buttons based on the Hough transform, structural inference, and multi-symbol recognition. Adbulla *et al.* [8] introduced an approach which uses color, shape, and size to search for buttons on the scene. However, such methods often fail when the lighting and background change.

The traditional method for button recognition, template matching [9], is used to rapidly identify the buttons. However, identification is a challenge because of the complicated background; the limited number of samples cannot represent all possible noisy situations in button recognition. However, some studies use machine learning to recognize the elevator buttons with approaches such as optical character recognition (OCR) [10] and CNNs [11]. However, high computational costs from CNN limits its application to mobile platforms.

C. OBJECT DETECTION BASED ON DEEP LEARNING

We propose a novel approach to detect the elevator buttons. In this study, we use only a single-shot detector to obtain the position of the target button. Then, we use visual tracking technology to follow the button.

Recently, deep learning has made a breakthrough in the field of object detection. Starting from region CNN (R-CNNs) [12], the two-stage approaches (finding objects' potential positions and detecting high-confidence candidates) and faster R-CNN [13], [14] have achieved exceptional detection quality. Another direction includes one-stage approaches such as You Only Look Once (YOLO) [15], in which researchers convert the detection task into a regression task to improve detection speed. Moreover, single-shot multibox detector (SSD) [16] combines the advantages of faster R-CNN and YOLO, resulting in fast, accurate detection.

D. VISUAL TRACKING

Although the above methods are fast and robust with graphics processing units (GPU), their applications for a service robot are limited, because of mobile robots' rudimentary hardware. We aim to process images on the mobile platform in real time, and our method uses the tracking approach after single-shot detection to locate the target button by comparing two neighbor frames. In earlier research, optical flow [17] and kernels [18] have been used to continuously compare two frames following a target. However, these methods are sensitive to continuous error accumulation. Current state-of-the-art approaches can be divided into three trends. The first trend is tracking-by-detecting: Zhang *et al.* [19] and [20] estimate the object's state with an established model. Based on this, an appearance model is combined with a detecting and updating procedure such as Kalal *et al.* [21]. Additionally, Hare *et al.* [22] have succeed in a long-term tracking task. Another method is to use a part-based method such as that of Nebehay and Pflugfelder [23], which tracks each part of the object separately, instead of tracking the complete object.

In contrast to earlier approaches, ours converts the task of elevator button localization into a combined detection and tracking structure. We demonstrate that this provides rapid, robust, and accurate button localization that can adapt to different environments. Furthermore, our approach is deployed on a mobile robot platform, and combined with laser SLAM technology to facilitate multi-story autonomous navigation

III. PROPOSED METHOD

A. OVERVIEW

Existing approaches to elevator button localization focus on detection by using a predesigned model. Unlike these ideas, our approach takes advantage of both detection and visual tracking, which are used to introduce a more adaptive localization approach. In our design, the CNN is used as the first shot on the scene to obtain an initial precise position of the target button. After that, our framework outputs the initial shot to the tracker. The tracker employs the pairwise consensus to locate the button by a vote of the feature points. To ensure that the tracking model overcomes challenges with the scale, rotation of the target, and background, we use a PN model to validate the tracker's results and update its model. To deploy the localization and mapping feature in the real robot, we use a laser SLAM method called cartographer mapping [6]. The well-known A$*$ algorithm is used to plan the shortest path on the occupancy grid constructed by the SLAM algorithm.

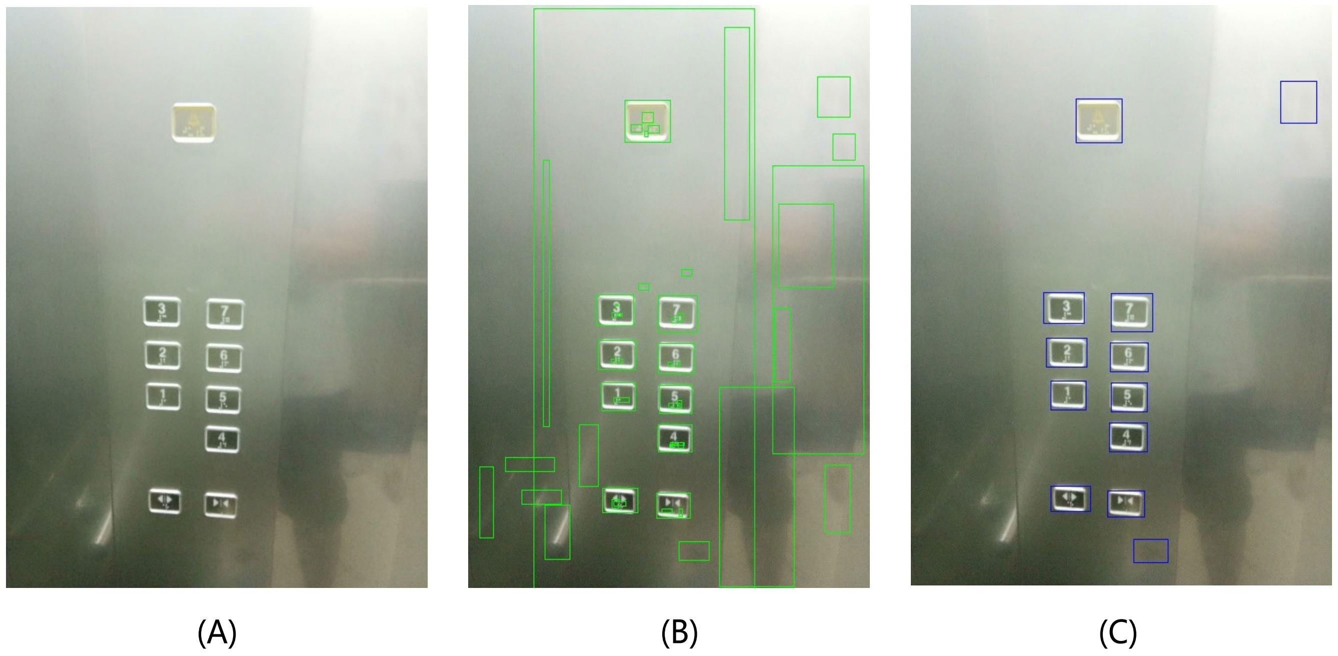

FIGURE 1. **Button localization results inside the elevator. (A) Original image. (B) Proposed contour extraction. (C) Effect after size filter and NMS.**

B. BUTTON DETECTION

1) ROI EXTRACTION

The first part of button detection with deep learning is to calculate the button's potential positions on the scene. Because most existing methods of object detection based on deep learning need to scan the whole scene and input several useless candidates into the neural network, the computational costs are high. As a result, we propose a classical image processing procedure to obtain the candidates for the buttons' proposed positions. Assuming that the buttons are always rectangle with clear contour, we propose a contour feature to detect their potential positions.

Our method includes three steps. First, the RGB image is converted into a greyscale image. To filter noise from the scene, we use the morphological opening operation (eroding and dilating the image) and median filter. Afterwards, we use the canny edge detector [27] to extract any edges that are considered candidates. The greyscale image conversion from RGB, morphological opening operation and median filter allows our method to filter low-frequency noises. The boundary of the elevator button is strongly featured. And, the buttons are generally manufactured by the light reflection material. Therefore, the canny scanning can constantly obtain the boundary of the buttons. In the experiment part, we design two data sequences to show that our proposed method can operate against the illumination challenges. In Fig 14, it was shown that our proposed method overcomes extremely challenging illumination condition. However, the illumination condition might change when inside the elevator, which leads to either high exposure or extremely dark images captured by the camera. This situation generally causes the failure of the button recognition. To solve this problem, an updated pose between the panel and the robot is required through slight rotation and movement of robot. A slight position and rotation change can greatly influence the illumination condition for the camera. The adjustment is operating continuously until the desired button is recognized.

As shown in Fig. 1 (B), false positives are still produced by the detector because the background objects create noise. As the sizes and shapes of elevator buttons are similar, we use a size controller (i.e., if the width/height ratio exceeds 1:2.5, or the length of the button is less than 10 pixels) to remove abnormally sized proposals. Moreover, a simple binary classifier and non-maximum suppression (NMS) [28] are applied to remove false positive noise and overlapping candidate regions. Fig. 1 (C) shows that all buttons are found when using our approach. The proposals are estimated according to followed formula:

$$\boldsymbol{P}_{x,y,w,h} = \boldsymbol{NMS}\left[(\boldsymbol{x},\boldsymbol{y}) + (\boldsymbol{w},\boldsymbol{h})\right] \quad \boldsymbol{if}\ (\boldsymbol{w}:\boldsymbol{h}) < \mathbf{1}:\mathbf{2.5} \tag{1}$$

where $\boldsymbol{P}_{x,y,w,h}$ denotes button proposals combining x, y coordinates and weight w and height h. The proposals are filtered by the NMS and size controller. Regarding about the circle button localization, [7] shows that possibility of using traditional circle detector to detect the target buttons in circle. Therefore, our proposed method can be simply modified to from rectangle detector to the circle detector to fulfil the requirement of circle button detection.

2) BUTTON RECOGNITION

After we obtained the potential positions of the buttons, the next step was to classify them. The standard approaches to object recognition use the OCR algorithm [10], but such

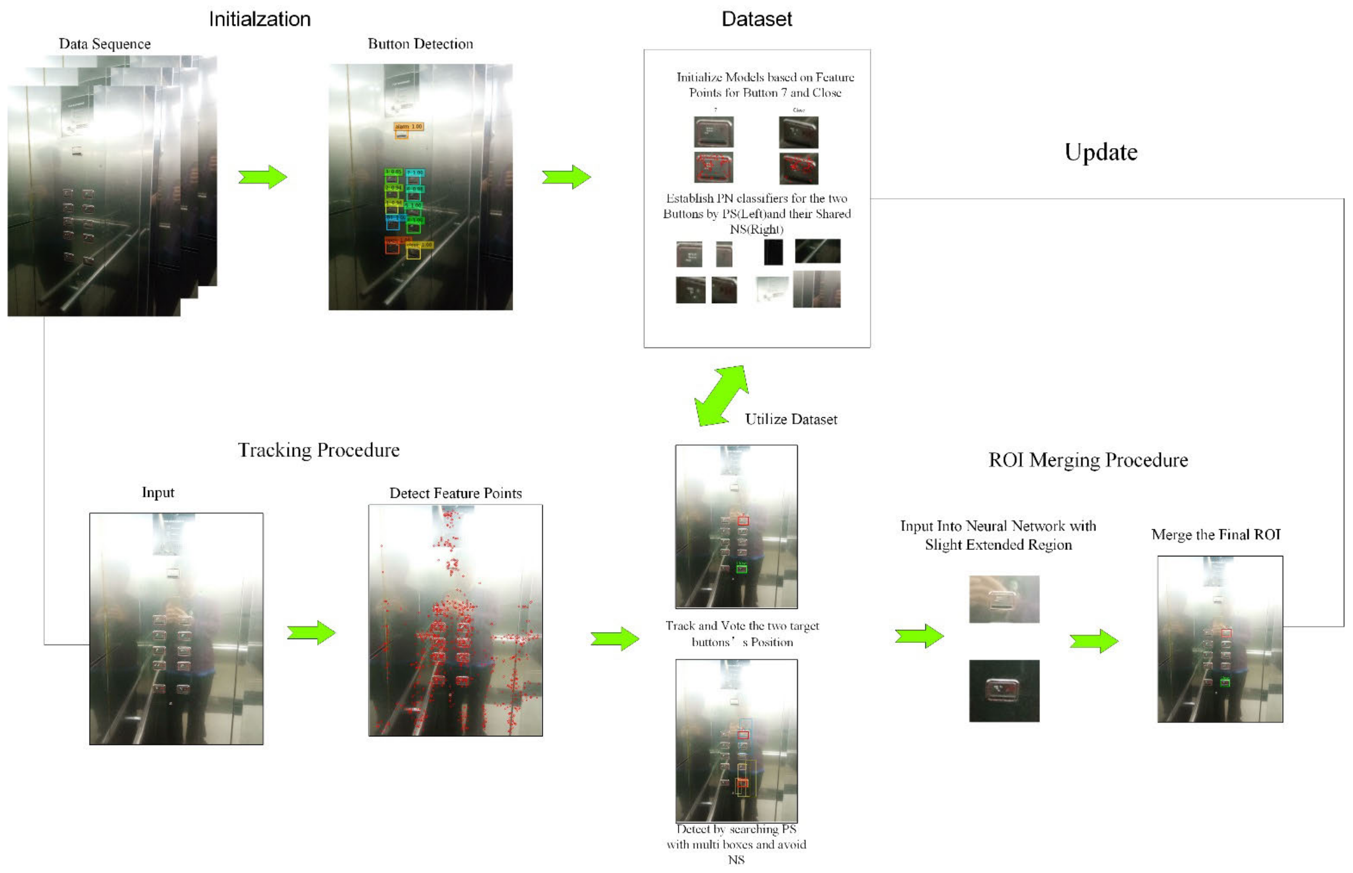
Initialzation
Data Sequence
Button Detection
Dataset
Initialize Models based on Feature Points for Button 7 and Close
Establish PN classifiers for the two Buttons by PS(Left)and their Shared NS(Right)
Update
Utilize Dataset
Tracking Procedure
Input
Detect Feature Points
Track and Vote the two target buttons' s Position
Detect by searching PS with multi boxes and avoid NS
ROI Merging Procedure
Input Into Neural Network with Slight Extended Region
Merge the Final ROI

FIGURE 2. Button tracking framework.

methods require initial image binarization and segmentation, which diminishes accuracy for complicated backgrounds. To maximize accuracy, our approach uses deep learning—specifically, SSD—to train the model and detect the target buttons. One difference between our version and the original version is the ROI extraction. We use the selective proposal method, as shown in ROI proposals, instead of global scanning, i.e., we employ the results of ROI proposals as input to the deep neural network. However, some of the target buttons' coordinate errors remain, and the SSD input will be extended by a factor of 1.5 from the original ROI extractions. To expedite the SSD, we use the MobileNet [31] as the backnet, and a depthwise convolutional layer instead of the traditional convolutional layer. Furthermore, we reduce the SSD input resolution from 300 × 300 to 128 × 128 to minimize computational costs. The final formula of our proposed method is shown as follow:

$$O^n_{x,y,w,h} = SSD\left(p^n_{x,y,w*1.5,h*1.5}\right) \tag{2}$$

where each output $O^n_{x,y,w,h}$ is recognized by SSD through inputting extended proposals $p^n_{x,y,w*1.5,h*1.5}$. The architecture of it is shown as in fig 3.

In each extra feature layer, there are 512 convolutional filters to allow the sufficient strength for recognition. The anchor factor is set as 1:1 and 1:1.5 to help localize the button more accurately. In each scale, 100 estimations are proposed to obtain the region of the targets. Through this architecture, the mobile SSD enables to rapidly recognize the buttons proposed by the ROI extraction without the thousands of proposal steps present in the original version. Due to the ROI proposals through button extraction helping the DNN to localize the targets with rough positions, and the SSD just requires to detect the position of the objects without global searching. In this way, we can maximize the detection accuracy and reduce the computing cost.. Hence, the miss-recognition and ROI flash are inevitable during the robot movement even when using an object detection DNN. The continuous localization is also important for the navigation and robot arm control. Therefore, the method to allow rapid,

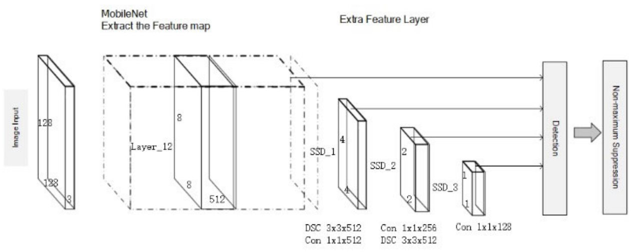

FIGURE 3. Architecture of mobile SSD.

accurate and low-cost localization of the target buttons is important..

C. VISUAL TRACKING

1) INITIALIZATION

The button tracking framework is shown in Fig. 2. After the detecting procedure outputs the objects' initial position, the visual tracking module uses the resulting target button ROI to initialize the tracker model. As our approach is based on feature points, it employs the features from an accelerated segment test (FAST) [24] to detect the feature points, and the binary robust invariant scalable keypoints (BRISK) [25] as the descriptor. With this combination, FAST allows rapid detection of the scene's corner information. Additionally, BRISK provides a binary descriptor to overcome scale challenges, as is required for the tracking task.

Our approach divides the image region into two parts: the foreground and background regions (FR and BR). The feature points inside the region are saved as the foreground feature points, and those outside of the region are saved as the background feature points. Once the groups have been distributed, the relationship between the foreground points (i.e., rotation of the target and distance) is saved, as detailed in the following section. When the results are obtained, a dataset is built including the properties (positions, rotation of the target, and distance) of the feature points. This dataset is then used in the tracking procedure. At the same time, our approach divides the image into small patches and saves them as positive and negative samples, according to region. By using these samples, our approach establishes a constraint model based on randomized decision forests [29] to record the environmental features in each tracking task. This improves the environmental adaptability of our tracking approach.

2) TRACKING

After the initialization, the original model is set up for the tracker. Next, the tracker starts building the correspondence between the previous frame and the current frame. Lucas–Kanade (LK) optical flow [17] is used in this part to track the points' trajectory, and to filter outlier points in the current frame, as determined by their position in the previous frame. The optical flow roughly builds the relationship between the two frames, and then a more detailed correspondence is built by global matching. The global matching function is:

$$d\left(p_i^{t-1}, p_j^t\right) < \theta \wedge \frac{d\left(p_i^{t-1}, p_j^t\right)}{d\left(p_i^{t-1}, p_k^t\right)} < \gamma, j \neq k, \tag{3}$$

where p_i^{t-1} denotes the feature points in the tracker's model, p_j^t are the matched points in the current frame; θ is the threshold of the translations of the matched feature points and γ threshold of the LK optical flow, which are set as 15 pixels and 20 pixels in our experiment. The p in the current frame is matched with those filtered by the LK optical flow; thus, the correspondence between two continuous frames is obtained. Brute force is employed as the matching mode in the global matching. In our case, the optical flow contributes to filtering the outliers to reduce computational costs, and the brute force provides a robustly similar matching to maintain correspondence accuracy. As a result, an efficient matching task identifies the closest feature point.

Afterwards, the object's appearance changes from the previous frame to the current frame are estimated according to the correspondence established by the matching step. The function of the translation between each two points is:

$$D\left(m_i, m_j\right) = \left\| \left(p_i^t - Hp_i^{t-1}\right) - \left(p_i^t - Hp_j^{t-1}\right) \right\|. \tag{4}$$

The Euclidean distance between every two matched points m_i and m_j is computed by a similarity transform H matrix composed of the rotation of the target and scale change. The scale change is calculated as follows:

$$s = med\left(\left\{\frac{\left\|p_i^t - p_j^t\right\|}{\left\|p_i^{t-1} - p_j^{t-1}\right\|}\right\}, i \neq j\right) \tag{5}$$

The scale ratio s is the median ratio of all pairwise points' distance changes between two continuous frames. In the end, the rotation change is computed by the equation:

$$r = \text{med}\left(\left\{\arctan\left(p_i^{t-1} - p_j^{t-1}\right) - \arctan\left(p_i^t - p_j^t\right), i \neq j\right\}\right) \tag{6}$$

Rotation change r is computed by the median value of arctangent changes for each pair between two frames.

Combining the distance, scale, and rotation changes, an ROI is voted by the center of the cluster as follows:

$$c = \frac{1}{|M|} \sum_{m_t \in M} \left(p_i^t - Hp_i^{t-1}\right), \tag{7}$$

where M denotes the class of all the matched points. The center of the target button is voted based on the average value of the coordinates of all the feature points' pairs. Accordingly, the similarity transform (c, s, r) is used to calculate the ROI's oriented position, thus completing the tracking procedure. The matching step builds the relationship between the two frames and the rotation/scale estimations to present the appearance of change in the object. Based on this, the final voting outputs a scale and rotation invariant tracking result.

3) VALIDATION, UPDATING AND REDETECTION

Change in the scale, lighting, blurriness, and rotation of the target is inevitable during long-term visual tracking tasks in button localization. Thus, a model-updating procedure is employed to maintain the model's robustness. In our approach, the PN classifier is used to detect the correct button position on the scene to validate the tracking. An online learning model is built by using every incoming frame according to the tracker's results. The purpose of this is to maintain the robustness of background changes. Subsequently, a multiscale sliding window scanner is employed to scan the patches and input them into the randomized-decision forest model in the last frames. The visualization of it is shown as follow:

The tracker outputs an initial position of the button, and an extended region is proposed. Apart from the GAN concept, the selective windows are employed to scan the potential position of the button which refers the history button information to the tracking module. In each box, the pairwise points are employed, which utilize the dataset saved in the random forest. As the tracking is a continuous procedure, and the change between nearby and current frames is insignificant, the previous points' pairs in nearby frames are compared with the current frame to vote the ideal position of the button under tracking. The visualization of the random forest is shown as follows.

The random forest is constructed by 20 decision trees. Each tree represents a frame in the nearest 20 frames. The depth of each decision tree is 15 and it has a single branch. Each tree represents 15 pairs of random points saved in its leaves to compare their intensity change to obtain a binary code. Each tree outputs a bayesian probability according to the formula P(F\C). F represents the result of each tree and C is the total number of possible sections, i.e., (2^15). Only two results (positive and negative) are saved in the histogram. And over 15 random decision trees output the positive result, the position will be regarded as the button's detected position after being merged with the tracker and GAN results. The size of the employed random decision tree is fixed, which avoids growing indefinitely.

The structural patches (x_i) of the given image are input into the forests, which then output a series of probabilities:

$$p_r(T \mid x_i) = \frac{ps}{ps + ns}. \tag{8}$$

The average value is composed of these probabilities, and our approach selects the window with the best score as the new ROI position. The detector's output is then compared with the tracker proposal. If at least 80% of the area overlaps, our approach selects the tracker's ROI; if not, the classifier and tracker's results are merged into a new ROI.

Because After the ROI output is received from the tracker and classifier, we employ the SSD (trained only by the shape of the buttons) to coordinate the ROI to restrict the tracker. The SSD inputs the appended region of the above ROI into the neural network to search for the most accurate button shape.

The SSD employed here is slightly different from the one which was introduced for recognizing the elevator button. As this block's main goal is simply to valid the shape of output, we further decrease the resolution of the mobile-SSD. The 128 × 128 inputs were reduced to 32 × 32, the convolutional depth from 512 to 64, down sampling only one time to keep the application real-time for visual tracking.

After that, the tracker's output, PN classifier, and SSD are merged by their median value to output the final ROI. For the tracker, the feature points inside the ROI are updated as

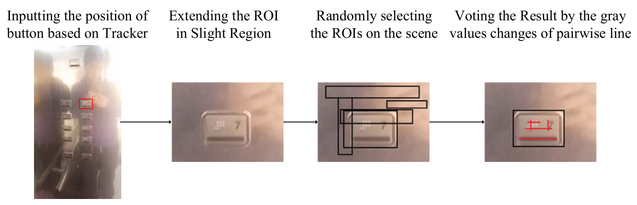
Inputting the position of button based on Tracker
Extending the ROI in Slight Region
Randomly selecting the ROIs on the scene
Voting the Result by the gray values changes of pairwise line

FIGURE 4. The framework of classifier. Red lines in the rightest image mean point pairs.

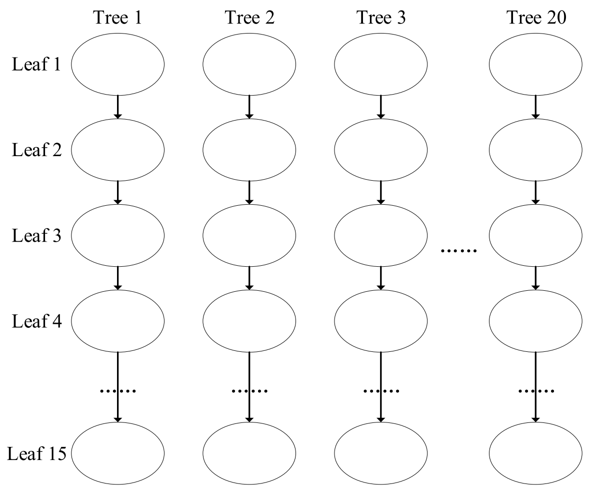

FIGURE 5. The architecture of the random forest.

new foreground points, and those outside as new background points.

A remaining issue is redetection after the loss of a button. In our approach, the PN classifier is first used to search the region for the lost buttons for 3 seconds. If it fails to locate them, the initialization step repeats.

D. GAN RESTRICTION

The generative adversarial networks (GAN) [34] employ the "GAME" theory which makes two neural networks compete with each other in every training step. In our proposed framework, a similar concept is employed to avoid the outlier of tracker caused by unsupervised learning. The outline of our proposed strategy is shown on figure 7.

The tracker performs as a generator to provide the rough position of the elevator button on the scene. The result of tracker and PN classifier is firstly fed into the SSD to detect whether the ROI still accurately localizes the elevator button (the system regards localizing loss if the confidence is less than 0.5). The discriminator then detects the button in a roughly extended region to adjust the button ROI to avoid potential ROI outlier. Then, the merging with PN classifier is operated to allow the tracker with history information from nearest 20 frames. The discriminator finally detects the button in an extended region to adjust the button's ROI to prevent potential ROI outlier happening in the upcoming frames. Its formula can be defined as follow:

$$X, Y, W, H = \max\left(D\left(G\left(x, y\right), w, h\right)\right) \tag{9}$$

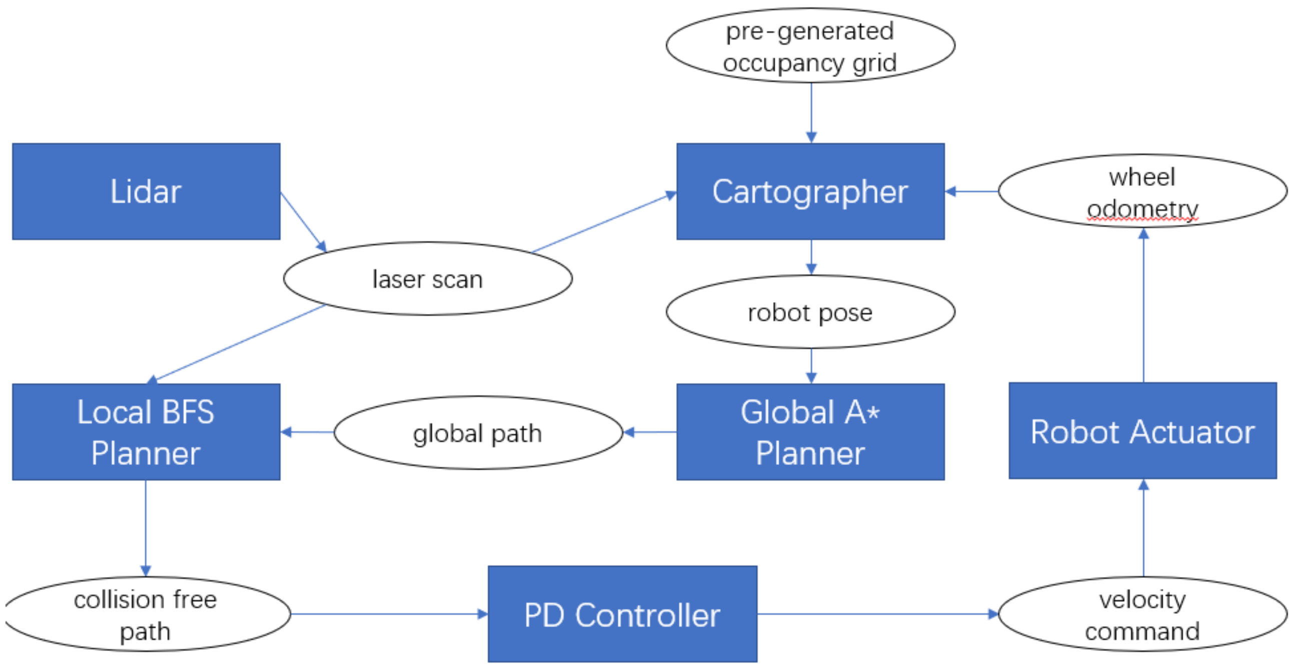

FIGURE 6. **Multi-story SLAM system framework.**

where the G (x, y) is the output of tracker, the w is the width and the h is the height of the ROI, the output are coordinate X and Y, and the width (W) and height (H) are calculated based on the maximal accuracy output from D. As we introduced, the visual tracking is an online learning procedure imposing no restriction on target types, and is sensitive to the background noises. The architecture allows the localization system to focus on the specific target (elevator button), which avoids common challenges both from object detection and visual tracking.

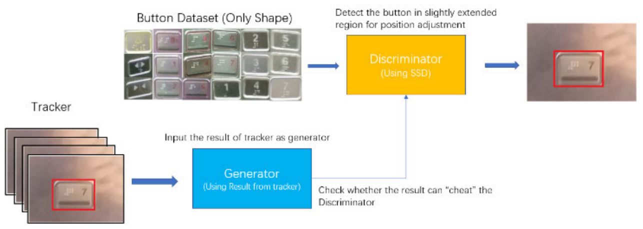
Button Dataset (Only Shape)
Detect the button in slightly extended region for position adjustment
Discriminator (Using SSD)
Tracker
Input the result of tracker as generator
Generator (Using Result from tracker)
Check whether the result can "cheat" the Discriminator

FIGURE 7. **Framework of generative adversarial neural network. The generator employs the result from tracker, and the SSD performs as discriminator.**

E. AUTONOMOUS NAVIGATION

To assess the efficiency of our proposed vision system, a prototype of the multi-story SLAM robot system was designed. The robot's autonomous navigation can be divided into three modules: a lidar SLAM module, a path planner and a proportional-derivative (PD) controller. A lidar is mounted on our robot to collect real-time depth data on the plane at its present altitude. With this laser scan, the cartographer mapping 2-dimensional SLAM method outputs the real-time robot position and incremental occupancy grid. For the sake of coordinate accordance, when the robot takes the elevator to another floor, the 2D coordinate system with its position on the former floor is replaced by an occupancy grid based on the button detection results. With the constructed occupancy grid, the A∗ algorithm identifies the resolution's shortest path from the current position to the target position. Based on the path, the PD controller generates a smooth linear and angular velocity output sequence for the robot actuator. The autonomous navigation system runs in real-time throughout our tests. The navigation system is shown in Fig. 6.

1) LIDAR SLAM AND PATH PLANNER

The cartographer employs the range data (lidar) which can obtain the real-time 2D point cloud at lidar altitude (height from ground to lidar) from the environment. However, an estimated map is essential for the robot's self-navigation. Thus, the cartographer pre-generates the map by manually navigating the robot through every floor.

The global planner, i.e., the A∗ algorithm, then generates a global path based on the robot's position and navigation target. However, exhaustive computing costs are incurred if the robot searches the global path of a sizable map. Therefore, a reasonable grid mapping manager plans the system's path. Unlike searching for a path on a single story, the costs of riding an elevator are difficult to determine. Because of this, dynamic path costs are difficult to compute, which can lead to failure. To solve this problem, we establish ''portals'' to connect different stories.

Fig. 8 shows three paths which change stories via a portal (i.e. an elevator). In our design, we establish a path with three steps. 1) Searching for the elevator nearest the robot

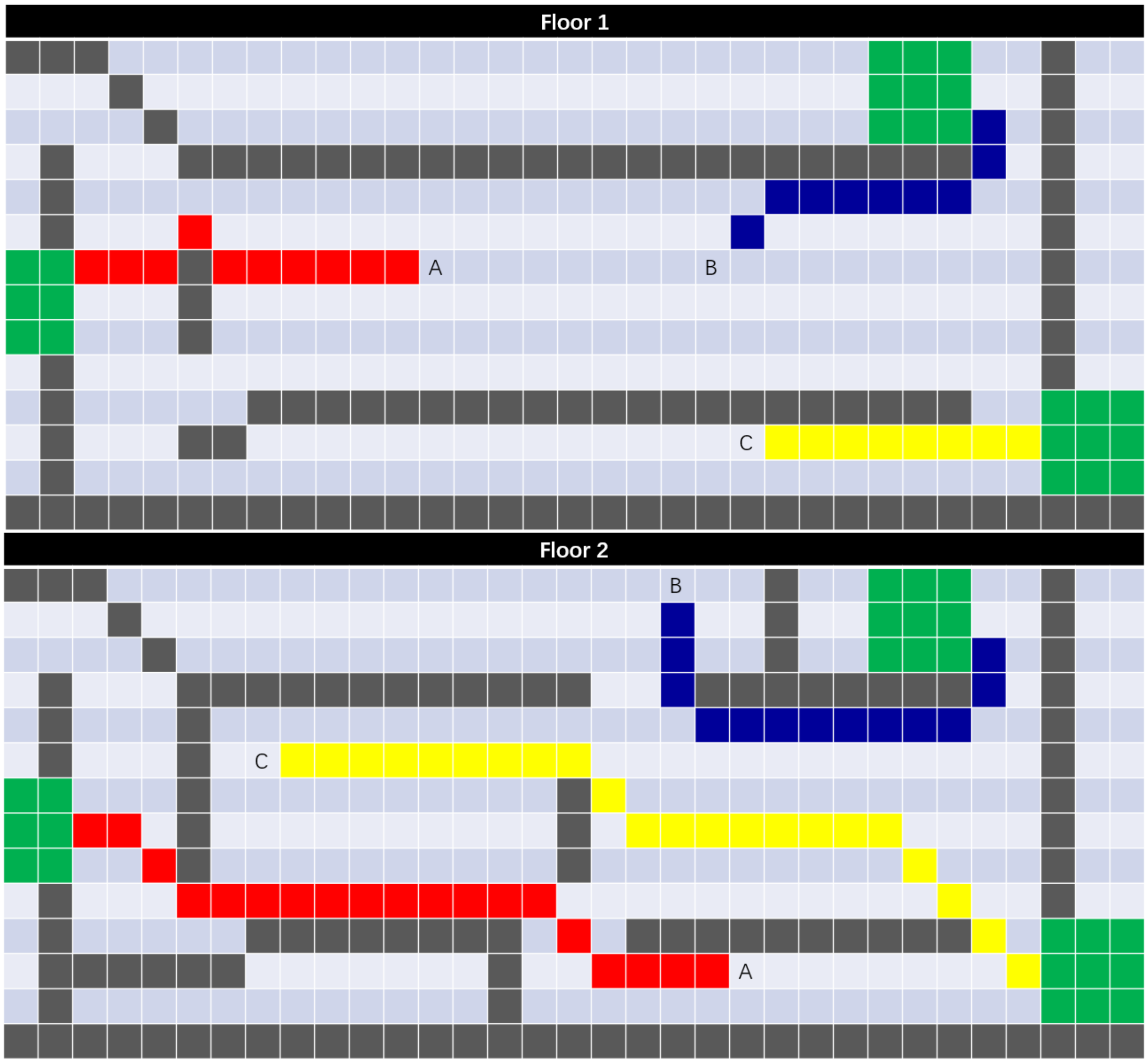

FIGURE 8. Visualization of multi-story path planning. Red, blue and yellow donate three trajectories which move from floor 1 to floor 2 via elevators (green). Gray denotes an unpassable area.

and moving to it; 2) Controlling the elevator and relocating to the target floor; 3) Setting the elevator as the initial position and searching for the path again. As A∗ is a path planner that uses heuristic distance cost, it is difficult to determine the costs of the path if the robot uses the elevator. However, because the task is conducted in a modern building, and the elevator's positions are fixed, we set them as the temperate destinations for the whole task. To search the nearest elevator, the Dijkstra's algorithm (first frame of A∗) is employed to search the path cost from current position to target elevators, then the nearest elevator is set as the temporary destination according to the cost on path planning.

Then, we use the control module to take the elevator to different floors. After the robot reaches the new floor, the initial robot position in the new floor is set with respect to the former robot position in the elevator, and A∗ globally searches the path to the destination.

2) LOCAL MAPPING AND PLANNER

As the global mapping only estimates the rough direction of this task, the local planner is essential for the robot to locomote and avoid obstacles. We employ a breadth-first search (BFS) path planner to locate a safe local path to an unblocked way point on the global path when dangerous obstacles are detected. The obtained local path with the

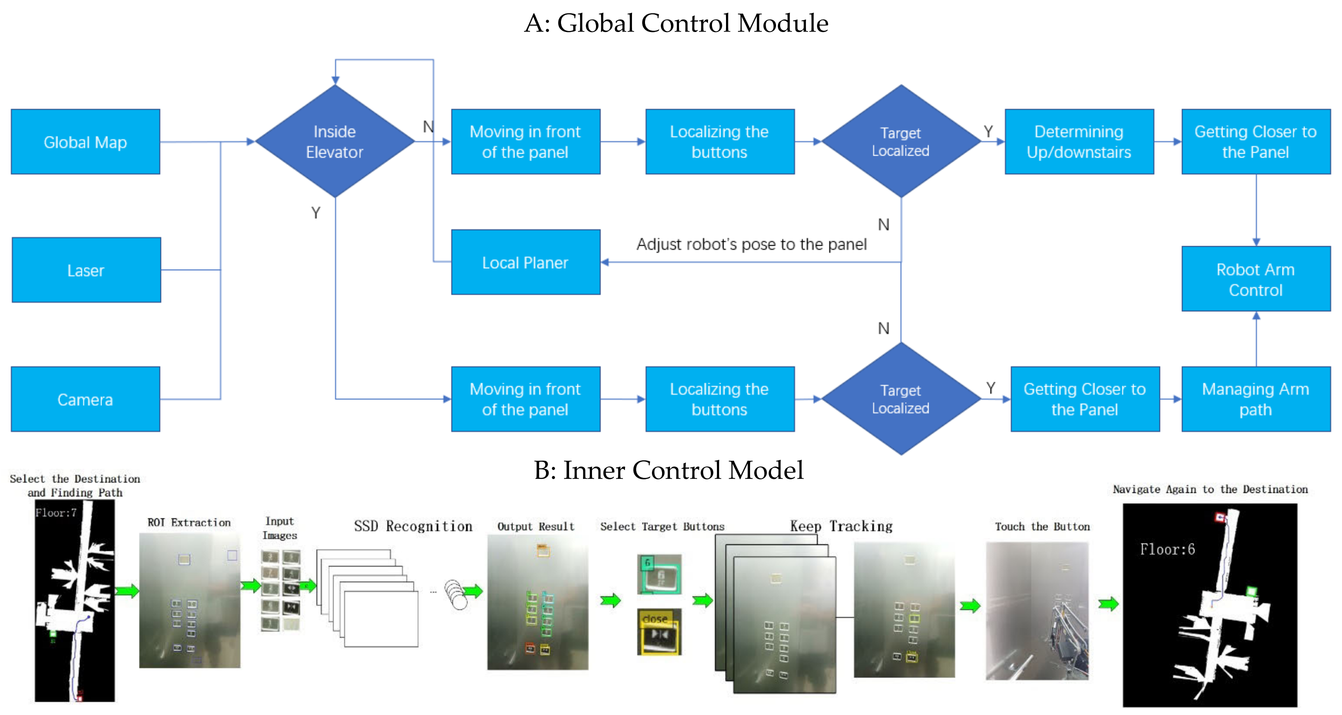

FIGURE 9. **Elevator controller module framework.**

residual global path is passed as a PD controller input for velocity command computing.

One challenge for the system is the pose drift that accumulates with the tracker. In other applications, it is difficult to determine the robot's position when the odometry is too big. The major reason for this is the inevitable accumulation of errors from the SLAM module, which is solved by the loop closure. During loop closure detection, the robot matches the current laser scanned map with the subgraph keyframes to split the accumulated errors into their appropriate subgraphs. Apart from utilizing the keyframes, the "portal" is also used as a strongly featured keyframe. The elevator is a perfect landmark for the SLAM framework because its geometry is fixed on the map. Therefore, the cartographer can easily conduct loop closure detection and bundle adjustment (BA) once the robot is inside the elevator, thus preventing pose drift accumulation.

3) LIDAR SLAM AND PATH PLANNER

Once the robot is near the elevator, the elevator control module is activated. See the flowchart in Fig. 9 (A).

The module is triggered if it is located inside the elevator on the global map. As the shape inside the elevator also has localization feature on global map, the position and pose of the robot on global map is directly utilized to determine whether the robot is inside the elevator. The size of elevator is 2.5 m $\times$ 2.8m, which is featured as a rectangle, and the resolution of the laser scanner is 1 ray /0.25 degree, thus allowing the robot to detect shape inside the elevator very accurately. Therefore, the localization inside the elevator is certainly correct.

Then, two branches are employed for the motion. If the robot is outside the elevator, it first moves to the position in front of the panel through the local planner. Then, it operates the button localization method to detect the "up/down" buttons. If a button is obtained, a decision is made whether the robot should go upstairs or downstairs by condition (target_floor – current_floor)>! 0. Once the target is obtained, the vision module continues tracking the button and approaches the panel. Once the distance (detected by the laser) between the robot and the panel is less than 0.3 m, the robot arm pushes the target button. During the waiting motion, the robot stands in front of the panel and uses the laser scanner to detect whether the elevator door is open, i.e., it detects whether the occupied grid ahead disappears in the local map. If the door opens, the robot enters the elevator and operates the elevator controlling module again. One major difference for the elevator to control the elevator panel outside/inside the elevator is the path planning. If it is outside, only one path should be generated to control the arm to press up/downstairs. If it is inside, the arm is required to follow the path to first press the target floor, then press the close button.

The framework of the elevator control module inside the elevator is shown in Fig. 9 (B). After the robot enters the elevator, it approaches the front of the panel via the local map. After that, it employs the detecting and tracking strategy to locate the target buttons. The illumination condition might change when inside the elevator, which leads to either high

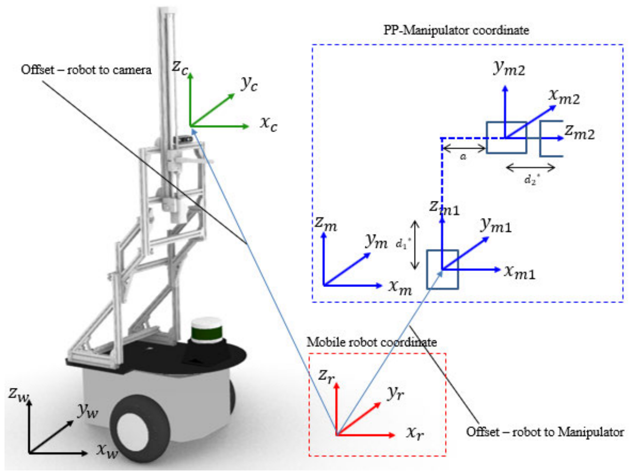

FIGURE 10. The design of robot and coordinate description.

exposure or extremely dark images captured by the camera. This situation generally causes the failure of the button recognition controller. To solve this problem, a new pose between the panel and the robot is required through slight rotation and movement of robot. A slight position and rotation change can greatly influence the illumination condition on the camera. The adjustment is done continuously until the desired button is recognized. Then, it approaches the panel and its arm pushes the button. There are two procedures for this step—the arm is programmed to move to the ‘‘floor’’ button and push it, and then move to the ‘‘close’’ button to close the elevator door. The waiting motion indicator is operated again, and it waits for the robot to reach the target floor. To obtain the level information, we installed several WIFI senders in front of elevator doors for each floor and used them to determine the story level based on the signal intensity and mac address of the senders. When the dbm of specific signal is between −30 to 0, the robot system regards the elevator reaching the target floor. As the maps on every story is prescanned by SLAM program and stored in the system. During the floor change, the map is updated by directly replacing the current map with the target map. In our design, the level information is obtained by wifi senders installed in front of every elevator door. The initial pose and coordinate for navigating in the target floor is inherited from the pose in the map of the previous floor. Nonetheless, during our experiments, the situation that elevator stopped on an unexpected level has not happened yet, because we conducted the experiment in an unmanned environment. After the robot arrives at the target floor, the global planning is operated again and the position of the robot inside the elevator is set as the original position.

4) ROBOT DESIGN AND ARM CONTROL

The design of our robot is shown in the figure 10. Our robot employs two wheels operation, which the rolls are concluded in the odometer. The laser scanner is installed on the board with 0.5m height from the ground. The bearing bracket is installed to allow the robot arm to operate in a sufficient

TABLE 1. DH parameter.

Joint N	θ_n (degree)	d_n (m)	a_n (m)	α_n (degree)
1	0	$d_1{}^*$	0	90
2	0	$0.0169 + d_2{}^*$	0	0

height to touch the elevator button. To avoid the laser to scan the bearing bracket as obstacles, we set the minimum range of laser with 0.2 m. In every frame, the vision module outputs the coordinate of the elevator button on the plane, which still requires to be further transformed into real-world 3D coordinates to operate the robot. The base coordinate is set by Z_w, Y_w, X_w. In between the base coordinate and the robot arm coordinate, the offset is added:

$$\begin{aligned} {}^r x_m &= 0.091m \\ {}^r y_m &= 0.065m \\ {}^r z_m &= 0.867m \end{aligned} \tag{10}$$

In the robot arm control, we utilize the two-joint robot arm to touch the elevator button. The Denavit–Hartenberg parameters (DH parameter) to describe the robot arm manipulator is determined as follow:

Our method presents a horizontal robot arm manipulator, in which the joint 1 performs for vertical movement, and the joint 2 performs for the horizontal movement. The homogeneous transform matrix is used to transform 3D coordinate, as follow:

$${}^0T_2 = \begin{pmatrix} 0 & 0 & 1 & a + d_2{}^* \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & d_1^* \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{11}$$

The final output can be summarized as inverse kinematics formula which is shown as follow:

$$\begin{aligned} x_m^* &= a + d_2{}^* \\ y_m^* &= 0 \\ z_m^* &= d_1^* \end{aligned} \tag{12}$$

Therefore, the x_m, y_m, z_m of robot arm coordinate converting to the robot arm through the inverse kinematic.

As the camera have a fixed position on the robot, the coordinate between underpan and camera can be converted simply by the offset:

$$\begin{aligned} {}^r x_c &= 0.035m \\ {}^r y_c &= 0.05m \\ {}^r z_c &= 1.036m \end{aligned} \tag{13}$$

As a result, we can convert the coordinate of the target buttons recognized by the camera to the coordinate of manip-

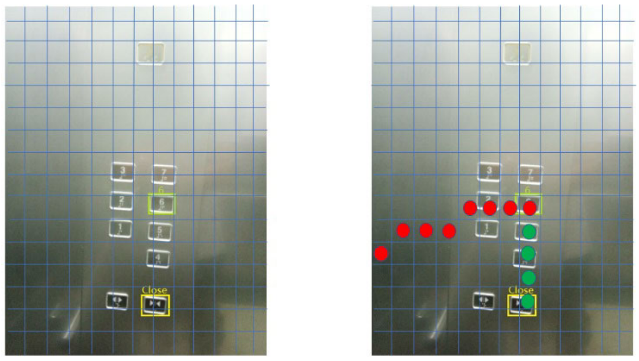

FIGURE 11. **The grid map on the panel and path visualization for robot arm inside the elevator. Red point means the first trajectory and green means the second.**

ulator target position:

$$\begin{aligned} x_m^* &= x_c^* + {}^r x_c - {}^r x_m \\ y_m^* &= y_c^* + {}^r y_c - {}^r y_m \\ z_m^* &= z_c^* + {}^r z_c - {}^r z_m \end{aligned} \tag{14}$$

While the vision module is processing, it returns the coordinate of the target buttons which are needed for the robot to navigate towards the panel. Every frame, the position of the elevator button is updated to the local planner. When the laser detects the distance between the panel and robot as less than 0.3m, the robot arm is operated to press the elevator buttons using the inverse kinematics solution.

The path planning architecture is shown in figure 11. When the robot has reached the active region, a grid map is generated for the panel. The y and z are obtained from the camera image and the x coordinate from the lidar. To calculate the target point of the robot arm, the camera and robot arm coordinates are calculated based on the mobile robot coordinate system, as shown in Formula (10) and Formula (13). Formula (14) allows to calculate the target coordinates x_m^*, y_m^*, and z_m^* of the robot arm. Formula (12) allows the calculation of the target points for each joint of the robot arm. The robot arm creates a trajectory to the target point, moves to the target button (6th floor), and presses it. The trajectory is based on the Dijkstra's algorithm path planning on grid map. Then, when the order is to close the elevator door, the target coordinates of the robot arm are changed, which creates the trajectory again, i.e. to manage the path from button (6) to button (close).

IV. EXPERIMENTS

We evaluated our approach in two scenarios: (a) the designed challenges inside the elevator, and (b) real robot operation. Our elevator button locating approach was first tested by the two video sequences which included scale, rotation of the target, exposure, and blurriness challenges. Both of our localization approaches—with and without a tracking strategy—were evaluated to emphasize the necessity of the tracking procedure. To validate our approach, we compared its results for three sequences with those of the state-of-art methods. Finally, a mobile platform was constructed on an own designed robot which used a 2DOI robot arm to touch the buttons. In our experiment, a laptop with a mobile CPU I7 7700 HQ, and 16G of DDR4 memory was selected to operate both video challenges and the multi-story SLAM system. The CPU I7 7700 HQ only had 2.8 GHZ basic frequency and could reach 3.8 GHZ with Intel Turbo Boost. This frequency was far less than that of the other CPUs employed by mini PCs (such as the IntelNuc series). A Real Sense was used to acquire the image data, and RPLIDARA3 was employed to obtain real-world range data. In the experiments, only the CPU was chosen to operate our system. The FPS was employed as the speed evaluating factor. Our combined detecting and tracking strategy achieved approximately 40 FPS on average, and the button locating procedure had 0.1 second detection duration. Thus, our system works in real-time for multi-story navigation, and it allows the robot to navigate between stories.

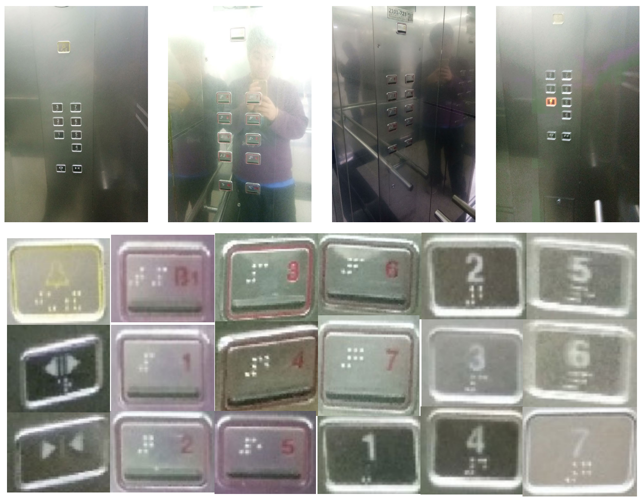

FIGURE 12. **Samples from the training dataset: the images have different sizes, illumination, blur rate, and angles.**

A. BUTTON LOCALIZATION

As various positions may appear while the service robot enters the elevator, a detection deep neural network is required to overcome the scale, rotation of the target, and lighting challenges. The deep neural network for button detection first needs to be trained. We trained the neural network on a desktop computer with a 1080 Ti GPU.

1) DATASET

To apply deep learning to object detection and recognition, a rich dataset is essential for model training. Since there are few public elevator button datasets online, we collected our own dataset by recording videos inside two elevators, with different angles and distances. Samples are shown in Fig. 12.

Then, the video frames were extracted as image data. To train the model, we collected 400 images with different angles and distances for each elevator's inner panel, and 200 images for their outside panels. To limit the influence of lighting, we augmented the data by changing each

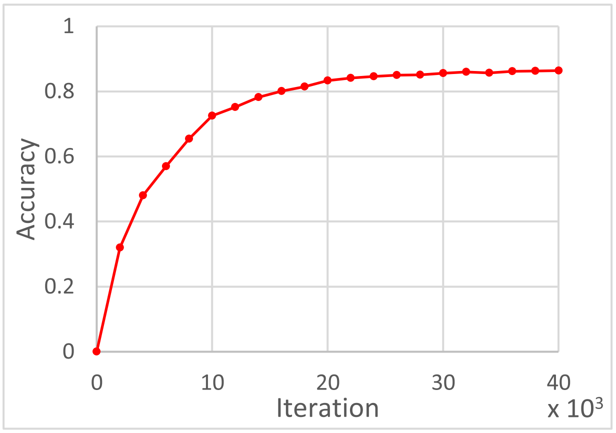

FIGURE 13. Accuracy vs iteration.

TABLE 2. Button detection performance.

Degree\method	Ours	SSD	Yolo	Faster RCNN
0-30 Degrees	92.3%	94.1%	92.9%	94.9%
30-60 Degrees	86.5%	88.6%	87.0%	88.2%
60-90 Degrees	76.5%	78.0%	75.9%	76.0%
MaP	85.1%	86.9%	85.3%	86.4%
Elapsed Time(ms)	93.4	434.7	222.2	909.1
FPS	10.7	2.3	4.5	1.1

image's color saturation and blur level. After manual labeling, we divided the samples into the following classes: Floor: B1, 1, 2, 3, 4, 5, 6, 7, open, close, upstairs, downstairs, and alarm.

To evaluate our proposed method's button detection performance in an elevator environment, we collected another 200 images in the elevator with different positions. We utilized over 600 images for training, and collected an extra 200 images for testing. As the size of the dataset was limited, a validation dataset was not employed. For each 5,000 iterations, the neural network was tested by the testing dataset. During training, several data augmentation approaches (such as horizontal flips, random crops or scales, rotation of the target, and translation) were used to increase input data. As our proposed method was employed to operate in specific elevators, and thus the environment was fixed, the overfitting problem was not a concern. After over 35,000 training iterations, the CNN label recognition surpassed 85% accuracy. The training progress is shown in Fig. 13.

2) EVALUATING BUTTON DETECTION IN DIFFERENT POSITIONS

To demonstrate the innovation of our methods, we compared them to other state-of-art object detection deep neural networks that utilize one- or two-stage strategies. The comparison was done with the testing dataset, and results are shown in Table 1.

The results illustrate that our button detection module was faster than other state-of-the-art methods, and had similar accuracy. The elapsed time of our proposed method is around 93.4 ms to recognize the buttons, while the other methods cost at least two times more. According to the table 2, the improvement on speed of our proposed network is significant when compared with other methods. This demonstrates the utility of our detection module for elevator button detection.

TABLE 3. Localization performance.

		WT	WOT	DP	SVM	OCR
Recall	DT1	0.95	0.90	0.85	0.15	0.14
Rate (%)	DT2	0.97	0.87	0.84	0.11	0.12
Average	DT1	40.4	10.1	3.6	70.2	65.6
FPS	DT2	39.5	10.3	3.5	72.4	67.2

3) EVALUATING BUTTON LOCALIZATION WITH OTHER METHODS

To further evaluate our framework, we compare our framework with other existing methods in our designed dynamic scenes. Because multiple challenges may occur simultaneously, we recorded video of fixed situations which may transpire during button localization. The two videos were recorded inside the two elevators in our building, with different lighting and different buttons.

The first sequence records an elevator with sufficient lighting. The extra challenge was the buttons' color. The elevator's background was white, which inhibited button localization because of the unclear edges between the background and the buttons. For the video, we first stood in the center of the elevator and adjusted the camera to it. Next, we rapidly moved to the front of the panel and then turned back. The motion in the second sequence was similar to that of the first one. The difference was the lighting, in which the background was dark, and the button edges were black. In the two sequences, blurriness, exposure, darkness, scale and rotation challenges were evaluated. We compared our approach with and without a tracking strategy (WT and WOT), deep learning method based AlexNet [11] (DP), support vector machines [26] (SVM), and optical character recognition [10] (OCR). Performance metrics were the recall rate and average FPS. The recall rate employs the percentage as unit and the FPS means frame per second. The DT means different data sequences.

As shown in Table 3 and Fig. 14, the methods that use deep learning (WT, WOT, and DP) achieved better button localization than traditional methods (SVM and OCR). However, the computational costs of deep learning methods were significantly higher. As shown in Fig. 14, WOT and DP usually fails when motion is blurry. This is because our training dataset lacks samples with blurry motion. Optimization makes our approach three times faster than DP; it is also slightly more accurate. Our detecting-cooperating with-tracking strategy overcomes the blurry motion; however, it has limited localization capability in the dark. This is because feature points cannot be detected in such situations. Because the number of feature points detected in sequences one and two was different, WT obtained a different average FPS, but they were both around 40 FPS. Apart from this deficiency, the tracking

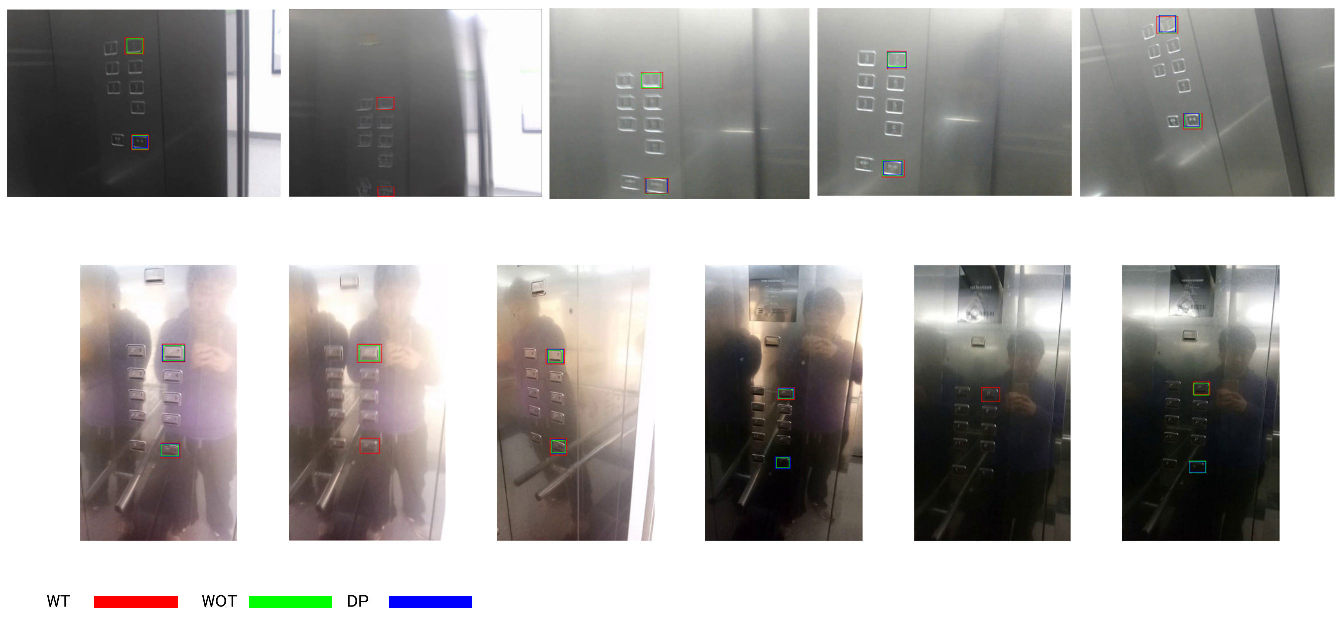

FIGURE 14. Results of button localization.

procedure provided a smooth ROI change and PN classifier. This allowed the model to adapt to the unknown environment and helped WT overcome the problems inherent to the two methods above. Thus, it obtained the highest accuracy with real-time performance.

The SVM and OCR both utilized the sliding window + classifier architecture based on a series of linear equations. However, the SVM and OCR generally perform well in tasks with minimal number classification. This is because the linear system's complexity is limited, especially in elevator button localization. Furthermore, the different buttons were similar, and this complicated both localization and recognition. One concern is whether a GPU can augment FPS. Those methods based entirely on deep neural networks will see marked improvement, while those based on traditional methods will only experience limited improvement. The reason for this is that GPUs perform well at float and parallel computation, but the response time through the bus of a peripheral component interconnect express (PCIE) unit takes longer than direct processing on the board. The deep neural network employs many neural nodes which requires parallel and backward processing to occupy millions of float computing steps. However, most of the traditional methods do not require such frequent parallel and float computing. Thus, the FPS of deep neural network-based methods will significantly increase; those of traditional methods will also increase, but less so. Considering the average CPU and memory usage rate in the table 3, the WOT consumes almost all resources in our mobile platform which means that the WOT may greatly hinder the performance of other modules in our robot system. In contrast, the WT employs the benefit of visual tracking which has lower computational cost, while being more reliable to obtain the target buttons.

TABLE 4. CPU and memory utilization rate.

Item	CPU Use	Memory Use
Dataset1/WT	30.2%	15.8%
Dataset2/WT	31.0%	15.7%
Dataset1/WOT	95.3%	86.6%
Dataset2/WOT	94.3%	89.2%

These results demonstrate that our detecting-cooperating with-tracking strategy is not only fast enough, but also has reliable button localization. Our method can overcome the situation in extreme dark, exposure, reflection and blur with various rotation and distance which allows it to operate in the real world.

B. MULTI-STORY SLAM SYSTEM

As the system manages to operate in the office environment, some chairs or desks may appear in the map. Although the sizes of legs on the chairs and tables are very small, our system still enables to scan them by the laser. The RPLIDARA3 employed in our experiment has angel resolution of 1 ray / 0.25 degree. This can be illustrated as follows:

Where max_gap = r * angle_in_rad (2m * 0.25 / 180 * pi = 0.87cm). The laser scans the plane in a specific height, where the x. y information of obstacles can be obtained. In the distance of 2m, the max gap is 0.87 cm, which is smaller than the width of most table and chair legs. As the buffer size of each obstacle's ray is set as 0.15m, the gap between legs will be regarded as impassable terrain, which avoids the situation where the robot tries to navigate through the gap between legs.

As shown in the map in Fig. 15, we first set a rough original position in the building and a destination position for the

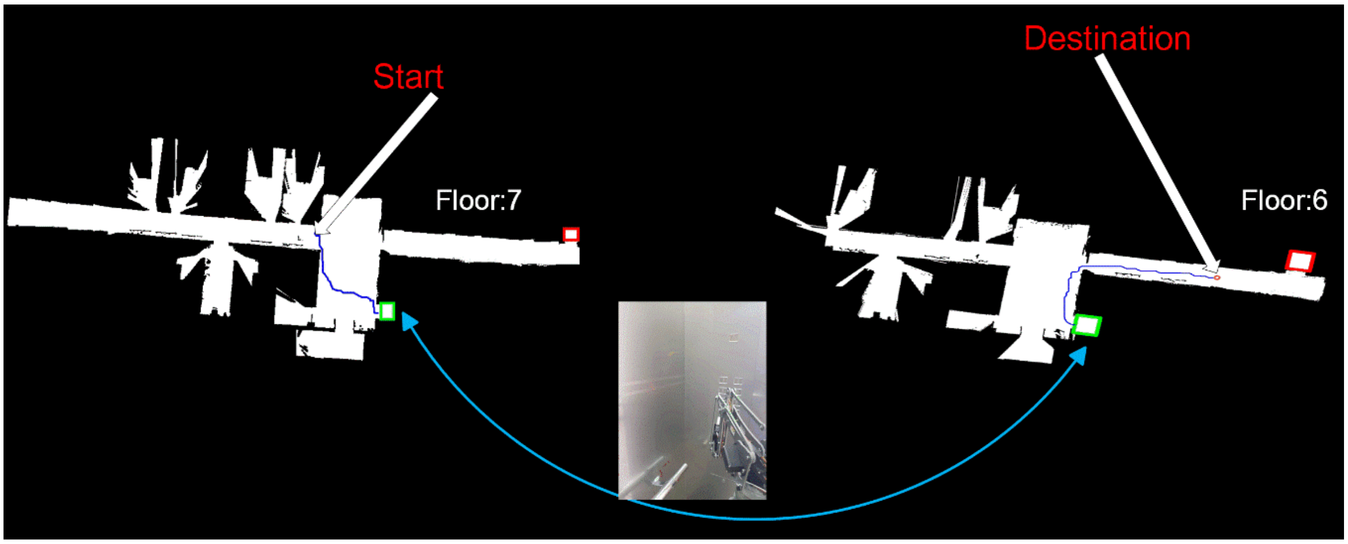

FIGURE 15. Robot's traveling path.

mobile robot. Then, the SLAM method navigated itself. After the initial position was chosen, our system chooses the nearest path to the destination (including the elevator). Once the elevator was chosen, the robot went to the front of the selected elevator and employed the detecting-cooperating-tracking strategy to locate the elevator button, i.e., the "up/down" button. Because the button localization accuracy in rotated and small-scale positions was sufficient, our approach identified the button panel for the initial localization. Combined with the robot movement, the robot went closer to the front of the panel. Once the scale of the button reached the threshold, the robot arm pressed the elevator button. After the elevator door opened, the robot entered the elevator and rotated itself to search for the target buttons. The operation was similar to that of pushing the elevator button; our robot tracks the target floor button and the "close" button, moves itself closer, and then pushes them. The final step is the initialization after floor change: once the target floor is reached, the occupancy grid is switched to the version with the new floor. The robot's initial position on the new floor is the elevator and its 2D position inside the elevator.

The results of this trajectory are shown in Fig. 15. The success of this trip demonstrates that our system overcomes the challenges of operating the elevator buttons, and thus accomplishes the multi-story SLAM task.

C. EXPERIMENT FOR CIRCLE BUTTON APPLICATION

To prove that our proposed method is able to be applied to different types of elevator buttons, an experiment on circle button localization was conducted. In this part, we collected another dataset of circle elevator button. The dataset consists of 200 images in the elevator with different angles of view and distances inside the elevator. An additional plan is proposed that the deep neural network is utilized to localize the elevator buttons. For deep neural network, 150 images are used for training and another 50 is used for testing.

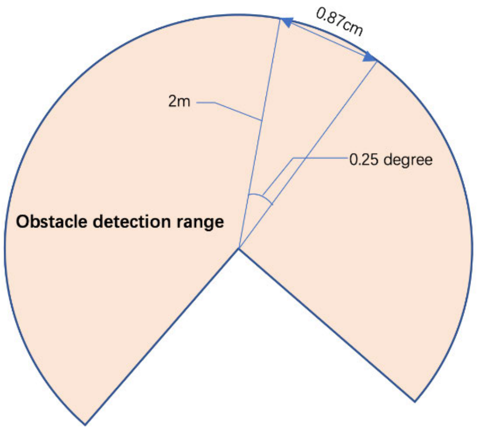

FIGURE 16. The calculation of min scanning gape between two rays.

TABLE 5. Recall rate of two methods in circle button detection.

Item	Recall Rate	FPS
HouGH	89.6%	245
Deep Learning	95.5%	5.8

The same test dataset is employed for both traditional method and deep neural network. There are two methods evaluated on button localization. 1.) The modified version of our proposed method, which employs HouGH [7] circle detection instead of rectangle detection. 2.) Mobile SSD, which is employed in the validation part of our method. The results and the visualization of our method are shown in table 3 and Fig. 17.

The traditional method based on hough features allows the detection in 89.6% recall rate with 245.0 FPS. The method based on mobile SSD obtained 95.5% recall rate with 5.8 FPS. The visualization of the two methods is shown.

On Figure 17, the result of HouGH detector is shown in the left side and the result of mobile SSD is shown in the right side. Most buttons are accurately extracted. The results

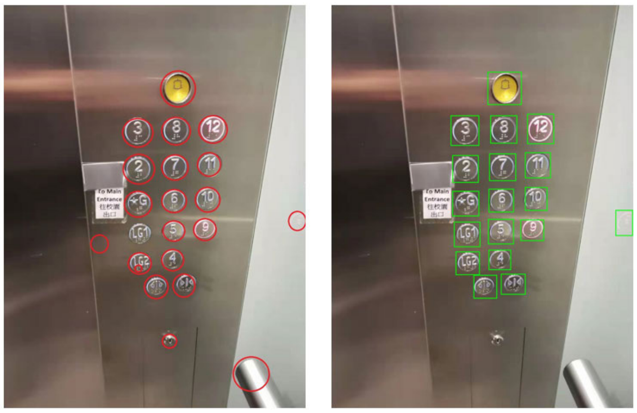

FIGURE 17. The result of HouGH detector is shown in the left side and the result of mobile SSD is shown in the right side.

in table 4 and figure corroborates that our method can be transferred to an environment with a circle button elevator.

V. CONCLUSION

This paper illustrates a novel approach for elevator button localization. Our approach takes advantage of the combined detection and tracking strategy which allows it to operate against the background challenges inside the elevator with real-time processing performance. Our approach detect and recognize the buttons in challenging environment with higher performance compared with other state-of-art methods. Combined with laser SLAM technology, our system runs throuhout the trip through a multi-story environment.

This work does have a limitation. The elevator button dataset cannot cover all makes of elevator, and thus users must collect and label several images when applying them to a new environment. In the future, we plan to extend the database and allow our approach to operate without expanding the dataset for every new environment.

REFERENCES

[1] J. Miura, K. Iwase, and Y. Shirai, "Interactive teaching of a mobile robot," presented at the IEEE Int. Conf. Robot. Automat., 2005, doi: 10.1109/ROBOT.2005.1570632.

[2] M. R. U. Saputra, A. Markham, and N. Trigoni, "Visual SLAM and structure from motion in dynamic environments: A survey," *J. ACM Comput. Surv.*, vol. 51, no. 2, p. 37, 2018, doi: 10.1145/3177853.

[3] A. Doucet, N. D. Freitas, K. Murphy, and S. Russell, "Rao-blackwellised filtering for dynamic Bayesian networks," in *Proc. 16th Int. Conf. Uncertainty Artif. Intell.*, 2000, pp. 176–183.

[4] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE Trans. Robot.*, vol. 23, no. 1, pp. 34–46, Feb. 2007, doi: 10.1109/TRO.2006.889486.

[5] S. Kohlbrecher, O. V. Stryk, J. Meyer, and U. Klingauf, "A flexible and scalable SLAM system with full 3D motion estimation," presented at the IEEE Int. Symp. Saf., Secur., Rescue Robot., 2011, doi: 10.1109/SSRR.2011.6106777.

[6] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2D LIDAR SLAM," presented at the IEEE Int. Conf. Robot. Autom., 2016, doi: 10.1109/ICRA.2016.7487258.

[7] X. Yu, L. Dong, L. Li, and K. E. Hoe, "Lift-button detection and recognition for service robot in buildings," presented at the IEEE Int. Conf. Image Process., 2010, doi: 10.1109/ICIP.2009.5413667.

[8] A. A. Abdulla, H. Liu, N. Stoll, and K. Thurow, "A robust method for elevator operation in semi-outdoor environment for mobile robot transportation system in life science laboratories," presented at the IEEE Jubilee Int. Conf. Intell. Eng. Syst., 2016, doi: 10.1109/INES.2016.7555147.

[9] D. Troniak, J. Sattar, A. Gupta, J. J. Little, W. Chan, E. Calisgan, E. Croft, and M. V. D. Loos, "Charlie rides the elevator—Integrating vision, navigation and manipulation towards multi-floor robot locomotion," presented at the Int. Conf. Comput. Robot Vis., 2013, doi: 10.1109/CRV.2013.12.

[10] E. Klingbeil, B. Carpenter, O. Russakovsky, and A. Y. Ng, "Autonomous operation of novel elevators for robot navigation," presented at the IEEE Int. Conf. Robot. Autom., 2010, doi: 10.1109/ROBOT.2010.5509466.

[11] Z. Dong, B. D. Zhu, and Q.-H. Max Meng, "An autonomous elevator button recognition system based on convolutional neural networks," presented at the IEEE Int. Conf. Robot. Biomimetics, 2017, doi: 10.1109/ROBOT.2010.5509466.

[12] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," presented at the IEEE Int. Conf. Comput. Vis. Pattern Recognit., 2014, doi: 10.1109/CVPR.2014.81.

[13] R. Girshick, "Fast R-CNN," presented at the IEEE Int. Conf. Comput. Vis., 2015, doi: 10.1109/ICCV.2015.169.

[14] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards realtime object detection with region proposal networks," presented at the Int. Conf. Neural Inf. Process. Syst., 2015.

[15] J. Redmon, S. Divvala, R. Girshick, and Farhadi, "You only look once: Unified, real-time object detection," presented at the IEEE Int. Conf. Comput. Vis. Pattern Recognit., 2016, doi: 10.1109/CVPR.2016.91.

[16] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu, and A. C. Berg, "SSD: Single shot multibox detector," presented at the Int. Conf. Eur. Conf. Comput. Vis., 2016.

[17] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," presented at the Int. Joint Conf. Artif. Intell., 1981.

[18] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 5, pp. 564–577, May 2003, doi: 10.1109/TPAMI.2003.1195991.

[19] K. Zhang, L. Zhang, and M. H. Yang, "Real-time compressive tracking," presented at the IEEE Eur. Conf. Comput. Vis., 2012, vol. 7574, doi: 10.1007/978-3-642-33712-3_62.

[20] K. Zhang, L. Zhang, Q. Liu, D. Zhang, and M. H. Yang, "Fast tracking via dense spatio-temporal context learning," presented at the IEEE Eur. Conf. Comput. Vis., 2014, doi: 10.1109/ICInfA.2017.8078941.

[21] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 7, pp. 1409–1422, Jul. 2012, doi: 10.1109/TPAMI.2011.239.

[22] S. Hare, A. Saffari, and P. H. S. Torr, "Struck: Structured output tracking with kernels," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 10, pp. 2096–2109, Oct. 2016, doi: 10.1109/TPAMI.2015.2509974.

[23] G. Nebehay and R. Pflugfelder, "Clustering of static-adaptive correspondences for deformable object tracking," presented at the IEEE Int. Conf. Comput. Vis. Pattern Recognit., 2015, doi: 10.1109/CVPR.2015.7298895.

[24] E. Rosten, R. Porter, and T. Drummond, "Faster and better: A machine learning approach to corner detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 1, pp. 105–119, Jan. 2010, doi: 10.1109/TPAMI.2008.275.

[25] S. Leutenegger, M. Chili, and R. Y. Siegwart, "BRISK: Binary robust invariant scalable keypoints," presented at the IEEE Int. Conf. Comput. Vis., 2011, doi: 10.1109/ICCV.2011.6126542.

[26] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining Knowl. Discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[27] Y. Lecun, P. Haffner, L. Bottou, and Y. Bengio, "Object recognition with gradient-based learning," *Shape, Contour Grouping Comput. Vis.*, vol. 2, no. 2, pp. 121–167, 1999.

[28] R. Rothe, M. Guillaumin, and L. V. Gool, "Non-maximum suppression for object detection by passing messages between windows," in *Proc. Asian Conf. Comput. Vis.*, 2015, doi: 10.1007/978-3-319-16865-4_19.

[29] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 8, pp. 832–844, Aug. 1998, doi: 10.1109/34.709601.

[30] M. Veloso, J. Biswas, B. Coltin, S. Rosenthal, T. Kollar, C. Mericli, M. Samadi, S. Brandao, and R. Ventura, "Cobots: Collaborative robots servicing multi-floor buildings," presented at the IEEE/RSJ Int. Conf. Intell. Robots Syst., 2012, doi: 10.1109/IROS.2012.6386300.

[31] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*. [Online]. Available: https://arxiv.org/abs/1704.04861

[32] J. Luo and S. Qin, "A fast algorithm of slam based on combinatorial interval filters," *IEEE Access*, vol. 6, 2018, doi: 10.1109/ACCESS.2018.2838112.

[33] M. Quan, S. Piao, M. Tan, and S.-S. Huang, "Accurate monocular visual-inertial SLAM using a map-assisted EKF approach," *IEEE Access*, vol. 7, pp. 34289–34300, 2019, doi: 10.1109/ACCESS.2019.2904512.

[34] L. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.

SHENLU JIANG received the B.Eng. degree from the College of Information Technology, Shanghai Ocean University, in 2015. He is currently pursuing the Ph.D. degree with the School of Electrical and Electronics Engineering, Sungkyunkwan University, Suwon, South Korea. He has been with the School of Electrical and Electronics Engineering, Sungkyunkwan University. He is also a Research Assistant with the Department of LSGI, The Hong Kong Polytechnic University. His research interests include computer vision, robot vision, remote sensing, and deep learning.

WEI YAO received the B.S. degree in photogrammetry and remote sensing from Wuhan University, China, in 2003, and the Dipl.-Ing. (Univ.) degree in geodesy and geoinformation and Ph.D. degree from the Technische Universität München (TUM), in 2007 and 2010, respectively. Since 2007, he has been a Scientific Collaborator and a Lecturer with the Institute of Photogrammetry and Cartography, TUM. Since 2011, he has also been a Senior Scientist in the research cluster of computer vision, remote sensing and navigation at the Munich University of Applied Sciences. He joined The Hong Kong Polytechnic University as an Assistant Professor. In 2017, he was selected for National Thousand Young Talents Program of China. He has already published nearly 90 academic articles in refereed international journals and conferences. His main research interests include active remote sensing technology towards reconstruction and analysis of spatial-temporal behaviors of objects, image processing and analysis, machine learning, and related environmental and industrial applications. His research work was also funded by Bavarian Excellence program based on the Bavarian Elite Aid Act. He was a recipient of the Best Student Paper Award from 2009 IEEE/ISPRS Joint Event on Urban Remote Sensing. Meanwhile, he was named a winner of the Chinese Government Award for Outstanding Self-Financed Students Abroad, which is granted around the world across all disciplines. He was also a recipient of Best Presentation award of the International Symposium on Mobile Mapping 2013 and best paper awards of several IEEE/ISPRS conferences. Since 2016, he serves as a Co-Chair for ISPRS WG III/6.

MAN-SING WONG received the M.Phil. and Ph.D. degrees in remote sensing and GIS from the Department of Land Surveying and Geo-Informatics, The Hong Kong Polytechnic University, in 2005 and 2009, respectively. From 2006 to 2007, he was a Fulbright Junior Scholar with the Earth System Science Interdisciplinary Center, University of Maryland at College Park, College Park. He is currently the Official Site Manager for the NASA's AERONET Station, Hong Kong. He has been publishing about 90 SCI journal publications, since 2005. He has been working in various projects including the use of remote sensing to study urban heat island effect, urban environmental quality, landslides, vegetation and ecosystems, spectral mixture analysis, aerosol retrieval, air quality, and dust storm monitoring. In 2014, he has received an Early Career Award from the Hong Kong Research Grants Council and two Faculty Award for Outstanding Performance/Achievement Award in Teaching, in 2014 and 2016, respectively; a Dean's Award for Outstanding Achievement in Research Funding, in 2016; and a Departmental Outstanding Research Award, in 2017 in The Hong Kong Polytechnic University.

MENG HANG received the bachelor's degree from Hohai University, in 2015, and the master's degree from Beihang University, in 2018. He is currently working in the Vision Technology Department, Baidu, China. His research interests include robotics and computer vision.

ZHONGHUA HONG received the Ph.D. degrees in cartography and geographical information engineering from Tongji University, Shanghai, China, in 2014. He has been a Lecturer with the College of Information Technology, Shanghai Ocean University, since 2012. His research interests include 3D damage detection, coastal mapping, photogrammetry, GNSS-R, and deep learning.

EUN-JIN KIM received the B.S. degree from the Division of Electronic Engineering, Dong-Eui University of College of ICT, in 2017. He is currently pursuing the Ph.D. degree with the School of Electrical and Electronics Engineering, Sungkyunkwan University, Suwon, South Korea. He has been with the School of Electrical and Electronics Engineering, Sungkyunkwan University. His research interests include coverage path planning and robot design.

SUNG-HYEON JOO received the B.S. degree from the School of Electrical and Electronics Engineering, Sungkyunkwan University, Suwon, South Korea, in 2017, where he is currently pursuing the Ph.D. degree. He has been with the School of Electrical and Electronics Engineering, Sungkyunkwan University. His research interests include mobile robot navigation and semantic slam.

TAE-YONG KUC received the B.S. degree in control and instrumentation engineering rom Seoul National University, South Korea, in 1988, and the M.S. and Ph.D. degrees from the Pohang University of Science and Technology, South Korea, in 1990 and 1993, respectively. From April to August 1993, he worked as a Chief Research Engineer at the Precision Machinery Institute of Samsung Aerospace Company and from September 1993 to February 1995 as a Senior Lecturer with the Department of Electrical Engineering, Mokpo National University, South Korea. Since March 1995, he has been with the School of Electrical and Electronics Engineering, Sungkyunkwan University, Suwon, South Korea, where he is currently a Professor. His research interests include intelligent robotics, adaptive and learning control, and visual sensor processing for computer-aided control systems.

• • •