Robust Perception Skills for Autonomous Elevator Operation by Mobile Robots

Steffen Müller[1], Benedict Stephan[1], Tristan Müller[1] and Horst-Michael Gross[1]

***Abstract*— Autonomous mobile service robots with transportation tasks are often restricted to work on a single floor, since remote access to elevators is expensive to integrate for reasons of safety certification. Therefore, already ten years ago first robots have been enabled to use the human interface for riding an elevator. This requires a variety of perception and manipulation capabilities as well as social skills when it comes to interaction with other people who want to use the elevator too. We summarize the progress in solving the specific tasks of detecting and localizing the required buttons to press robustly. A deep-learning approach for detecting buttons in images is combined with a verification based on predefined knowledge on button arrangements in the elevator's control panels. Also perception of the elevator's state and our realization of the robot's elevator riding capabilities are discussed.**

I. INTRODUCTION

Riding an elevator is a subconscious action for human beings, even if there is a lot of time to think about it while waiting for the cabin. For a mobile service robot in contrast the process has to be decomposed into clearly defined steps each requiring specific recognition and articulation skills.

To reach a destination on another floor, a robot must first implement a path planner capable of managing multiple floors. Already in 2007 Kang et al. [1] described navigation skills necessary to plan on multi-story maps. In general, a combination of topological and metric planning allows to solve the planning problem [2].

Once the plan shows that a transition between floors is needed, the robot can go to the lift on its current floor, and then the elevator procedure starts.

There exists research about robots without manipulation skills, which need the assistance of humans for operating the lift. [3] as well as [4] described how a robot seeks a helping hand and analyzes the people's intent in helping the robot. Other social studies [5], [6] analyze the effect of a robot's appearance and interaction behavior in a conflict situation while waiting for an elevator.

On the technical side, an autonomous robot first needs to call the elevator by pressing the respective button. For that purpose, a robust detection and localization of the button in the 3D operational area of the robot is needed. In the early days, this has been achieved using classical approaches in image space. For example SIFT-feature-based detection or template matching [7] have been used. Recent solutions rely on neural-network-based detectors. Zhu et al. [8] trained a Faster RCNN detector on elevator buttons. In order to also handle situations with unconventional buttons, they combined the Faster RCNN-based region of interest (ROI) detection with optical character recognition (OCR) to read the button labels. Once a 2D proposal for a button location exists, the utilization of depth data either from 3D cameras or LIDAR sensors can be used to project the image position into 3D space. A button detection with a neural network in combination with a LIDAR for 3D coordinate transform can be found in [9].

The detection of individual buttons alone may fail or the wrong label might be predicted in a real world application where partial occlusions or other artifacts may disturb the perception. Therefore, the geometry and arrangement of the complete control panel has been used to disambiguate the meaning and position of individual buttons. Abdulla et al. [10] for example detect button panels and correct the exact location by using artificial landmarks at the corners of the panel. We adopted the idea of using knowledge of the complete control panel to make button localization more resilient.

Assuming that the correct 3D coordinate of the desired button could be found, the next step is to push the button. This is a straight forward task utilizing the motion planner for the robotic arm. To decide if the cabin has arrived and whether the robot can safely enter the lift, further perception skills are necessary. Lee et al. [11] recognize the state of the elevator by laser, and a neural-network-based classification of the arrow signs is used to confirm success of the push button action. Once the cabin is entered, the goal floor has to be selected by means of another button operation. Then the robot has to recognize the correct floor to leave the lift. This can either be done by means of artificial landmarks for localization on the goal floor [11], or the robot can rely on acceleration sensor tracking [12] and reading the information panel of the elevator [13]. In all cases, the robot needs to be specialized on the present elevator, or the environment has to be adapted to the robot's needs.

In our work, we tried to avoid any modification to the environment and used pragmatic solutions for controlling the whole sequence only with feedback on the lift state extracted from the SICK laser range scanner of our robot. By reducing the number of different image-based detection systems to a minimum, the potential points of failure should decrease as well, which makes the whole system more robust.

In Literature typically only solutions for the individual

*This research has received funding from the thuringian project of innovation potentials as part of the thurAI project (grant agreement 2021 FGI 0008).

[1]Authors are with the Neuroinformatics and Cognitive Robotics Lab, Technische Universität Ilmenau, 98693 Ilmenau, Germany steffen.mueller@tu-ilmenau.de

979-8-3503-0704-7/23/$31.00 ©2023 IEEE

in: European Conference on Mobile Robots (ECMR 2023), Coimbra, Portugal, IEEE 2023

Fig. 1: Robot platform *Zeus* as a combination of a SCITOS G5 base by MetraLabs GmbH with an arm by Kinova

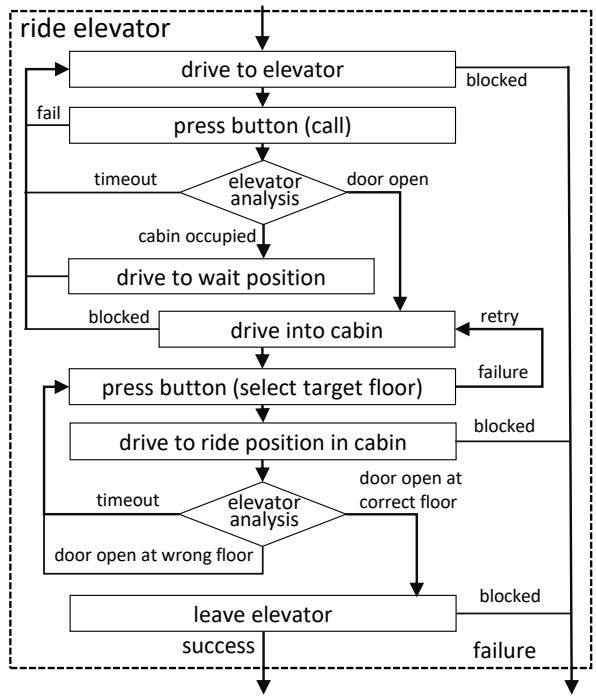

Fig. 2: Procedure of riding the elevator

skills of the robot can be found. In practical application there arise many challenges like short time spans to react when the lift doors open or unforeseen interactions with people occupying the robots way. We discuss these in the following as well.

II. Robot Platform and Background

This work is part of the research project RobInCare[1] dealing with basic capabilities of mobile service robots to enable autonomous navigation in nursing homes for the elderly. When deployed to deliver mail and other items or to pick up residents at their homes for group events, the robot has to open and close doors and needs to use elevators with its on-board manipulation skills.

The robot we use is a SCITOS G5 differential drive platform equipped with an additional Kinova Gen II 7 DoF arm for manipulation tasks. The wheels are big enough to avoid getting stuck in the 4cm gap at the lift cabin entrance. For perception of the environment, the robot has an Azure Kinect and an ASUS Xtion depth camera on top of a pan-tilt unit (PTU). Additionally, we use the three axes accelerometer of the SCITOS G5 for recognizing the vertical movement of the lift.

Navigation and localization of the robot platform relies on a SICK S300 laser range scanner mounted in a horizontal position. For self-localization in the environment, we apply a Monte Carlo Localization [14], which uses laser to map matching, and additional observations on button panel locations which have been mapped before. This will be explained in Sec. V-B in more detail.

The navigation skills rely on [15] which allows maneuvering in the lift cabin that is only a few centimeters wider than the robot's footprint. The planning of arm motions is done according to [16], which allows us to consider perceived obstacles in real-time.

III. Overview of the Elevator Riding Procedure

The whole elevator riding process is integrated in our behavior-based software architecture implemented in MIRA[2]. The higher order behavior decides to use the elevator when the plan to the goal requires changing the floor. Then the *ride elevator* procedure of Fig. 2 takes over. Here the robot at first navigates to a starting position in front of the elevator. Then the press *button procedure* is triggered which will be explained in Sec. IV. This returns control after the push force has been recognized and the robot's arm has been retreated to its home position allowing for safe navigation. An additional feedback on the success of the call button action is not implemented, since the robot will notice the effect when the elevator doors open within a reasonable time. So, the analysis of the elevator by means of the laser scan is continuously running in the background yielding information on the cabin door's state and the occupancy of the cabin. This is further explained in Sec. VI.

Once the door opened, the robot can enter the cabin if it is empty. Otherwise, it will give way for the people and retries the whole procedure. This is commented politely via specific voice outputs. In case that the robot reaches the goal position inside the lift cabin, again the *press button* procedure can take control. In our database, the semantics of the different buttons of the control panel have been mapped and, thus, the correct one can be selected according to the target floor. After

[1]https://www.robincare.de/

[2]https://www.mira-project.org

pressing the button, the robot places itself in front of the exit door, because the doors open for a short moment (6 sec) only. Again an explicit feedback on the success of the button action is not implemented. The robot will react on the opening door and counts floors to decide whether the target floor has been reached or an unexpected intermediate stop took place. If the later case, when the door is opening the robot comments to the people who have called the elevator that he is occupying the cabin and asks for patience. If there are no people waiting and the cabin is not going to move to the next stop, the button for the target floor is pressed again, which is also the case if the cabin does not start to move after the selection of the target. In the normal case, the robot simply drives to a point outside the cabin and control is returned to the higher level behavior responsible for the navigation to the actual target (e.g. the apartment of the resident to be visited). There are several possible points of failure, which in most cases can be handled by a retry. Only if the navigation path is blocked, or the whole system gets stuck after a collision, the procedure has to be aborted with an exception.

Challenges for the design of the procedure were the short time intervals for maneuvering the arm in a safe position for platform movements and the required time to plan the movements. If the elevator is already at the current floor when the robot presses the call button, he has only around six seconds to enter the cabin before the doors close again. This has been solved by finding an optimal position for the robot when pressing the button. It has to be nearly centered in front of the door, but on the other hand, this should not be too close to the door to allow for robust button panel perception and room for manipulation actions. Finally, the robot is slightly angled such that it can enter the cabin with a gentle arc movement. Additionally, the retreat motion of the robot arm could not be executed with the motion planner in the short time. Instead, we reversed the trajectory for the pushing action and executed that at a higher speed assuming that the scene is still free of obstacles in that region. The same timing issues are present when the elevator already is on the target floor, and the robot would presses the respective button inside, which causes the door to open immediately, and there is not enough time to leave the cabin. Luckily, this case only can happen when the path to the outside of the cabin is blocked by people and the robot has to retry to leave the cabin after verbal communication to the people outside. A solution for that problem is that if the robot is already on the goal floor the button for another floor is pressed, causing the elevator to take a detour, which gives time for maneuvering inside the cabin. The original target floor then is reached after another cycle of selecting the destination by button.

IV. Pressing the Button

The actual procedure of pressing a desired button is shown in Fig. 3. It starts with a navigation to a panel specific interaction position, which allows for a good camera view at the buttons as well as good reachability. Then the camera is pointed at the control panel, while button detection and panel localization run in background (see Sec. V-A). Once

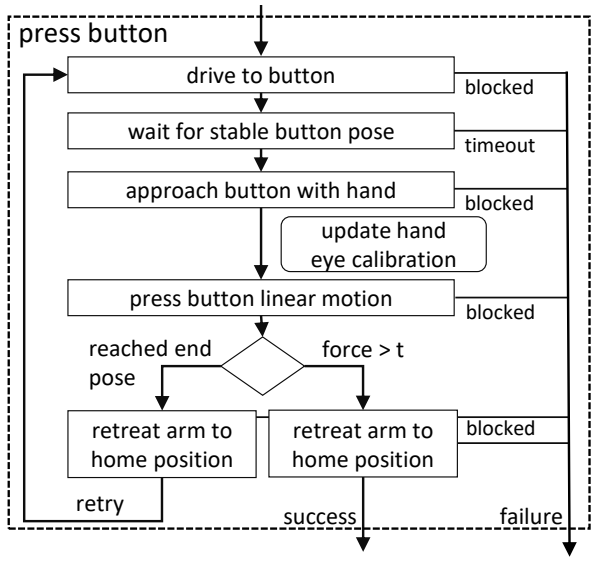

Fig. 3: Procedure of pressing a button

the pose estimation stabilized, the manipulation starts by bringing the robot's finger 10 cm in front of the estimated button. Since the target position of the button is estimated in camera coordinates, and the camera is on a PTU, and additionally the whole robot frame is not rigid, there is a calibration offset from the internal robot model and the actual position of the hand in respect to the camera. Depending on the position in the reachable area, this deviation can be up to 3cm, which is critical for hitting the button correctly. To solve that calibration problem, we included an estimation of the end-effector pose in camera coordinates by means of an ArUco marker [17] detection. A hexagonal marker arrangement has been mounted to the robot's wrist, allowing to see at least two non-co-planar markers at a time (see Fig. 4). By means of the calibrated camera, the 6D pose of the marker arrangement can be estimated, yielding an offset to the internal model of the robot. This offset then is used to correct the goal position for the next movement to be executed. This is a linear motion to a virtual point 2 cm behind the button of interest. During the execution, the collision check for the fingertip is disabled to allow for the contact with the button, which otherwise would be avoided because the wall is contained in the collision scene of the motion planner. The success of the operation is monitored by means of the end-effector force. If the force in movement direction exceeds a threshold t of 6 N, the button is supposed to be pressed and the arm can be retreated to the home position. If the force has not been noticed before the end point is reached, then the button is supposed to be missed and the whole process starts over again.

Possible points of failure for the whole procedure are unreachable positions. Either the position for the robot platform is occupied, or the robot's arm can not reach the desired start and end point of the push trajectory.

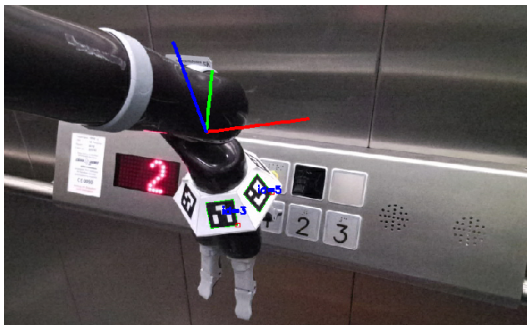

Fig. 4: Localization of the robot's end effector in camera coordinates by means of ArUco markers used for online hand-eye calibration.

V. Button Detection and Localization

Detecting and distinguishing individual buttons either with classical or machine learning methods always is prone to confusion of labels or complete misses due to occlusion for example (see Fig. 5). In order to make the button pressing process more robust, we decided to incorporate prior knowledge on the existing button arrangements in our operational environment. The raw detections of buttons are compared to previously mapped button panels in order to optimize their position and correct labels. The recording of panel configurations during installation is done using the same detector as for online recognition but under optimal recognition conditions without occlusions and from a perpendicular view.

Furthermore, to reduce the influence of occasional false detections, the recognition process during application is not only done once but runs in background at 4 Hz yielding a stream of button panel locations that are filtered for outliers by means of a geometric median.

On the first instance, the location of the button panel and therefore the exact position of the individual buttons is used for planning a push trajectory for the end effector of the robotic arm, but the deviation of the panel's relative position in robot coordinates to the mapped panel position is also fed into the MCL as additional cue for localization. This allows a reduction of the typical laser-based localization error from about 5 cm, which is related to the occupancy grid map resolution, to a value in the 1 cm range. Therefore, also static collision scenes can be considered by the robot during manipulation, which prevents contacts to walls and surfaces that are not visible in the depth camera directly.

In the following the realization of the neural-network-based button detection is described before the actual localization of button panels is explained, which uses the set of detected buttons.

A. Raw Button Detection

For the recognition of individual buttons in color images of our Azure Kinect, we trained a Faster-RCNN network on the dataset of [8].

The aim is to distinguish 16 classes of buttons, which are the special functions (open doors, close doors, alarm) as well as most prominent floors B (basement), L (lobby), G (ground floor), 1, ..., 9, and some wild card buttons with unrecognizable labels called 'button'. The exact semantics of the buttons is not necessarily to be detected by the network since it is mapped to the buttons in the button panel as stored in the robot's database as described in the next section.

The network architecture we used is more lightweight than in [8] since we use a ResNet-50 backbone instead of ResNet-101 in order to save resources on our mobile hardware (Nvidia Geforce RTX 2060). Additionally, we did not use the OCR-RCNN approach proposed in [8] as the number of floors and therefore the possible elevator buttons are within the set of 15 classes. The slightly worse AP_{50} value of 85.8% (compared to 90.1% of the ResNet-101) is acceptable due to the further processing in the panel matching process.

Once the bounding boxes of candidate buttons have been found, the next step is creating 6D poses for each of them by means of projecting the boxes onto a point cloud of a depth camera. We use the ASUS Xtion depth image due to more reliable geometric properties of that active stereo camera. The depth image of the Azure Kinect camera shows material dependent depth offsets and other artifacts due to the time of flight approach. From the 3D points inside the bounding boxes, the surface normal and the xyz-coordinate of the center can be extracted by means of a plane regression. Using the vertical axis as granted, the full rotation matrix can be easily computed for the normal and the z-unit vector by means of three cross product operations, which completes the 6D pose of the buttons.

B. Panel Matching

Having the incomplete set of button poses and respective predicted labels, the association to known button panels can be done. From the robot localization we can find the panel of interest in our database and for that the relative position of the buttons to the panel's origin (at its center) is known.

The detection of the button panel's pose in respect to the camera is done by a Maximum Consensus approach. For that, each pair of possible associated buttons, one from the known panel in the database p_i and one from the current detections d_j, is used to compute the relative 6D transformation between them. This in the following can be used to project all the detected buttons onto the known panel. For each detection, we then search for the closest button p^* on the known panel and accumulate the distances. Here the match of the predicted and the mapped label are taken into account. Label mismatch yields a penalty offset ρ to the distance causing a possible association to neighboring buttons that might be a better match.

From that accumulated minimal distances to associated buttons a matching score $m(p_i, d_j)$ is computed to rank the predicted transformation.

$$m(p_i, d_j) = e^{-\left(\sum_j min_i\left(|\mathbf{p}_i - \mathbf{d}_j| + l(p_i, d_j)\right)\right)^2/\sigma^2} \tag{1}$$

Here $\mathbf{p}_i$ and $\mathbf{d}_j$ are the xyz-coordinates of the panel buttons and projected detected buttons respectively. The term $l(p_i, d_j)$ has a value ρ (free parameter) if the labels do

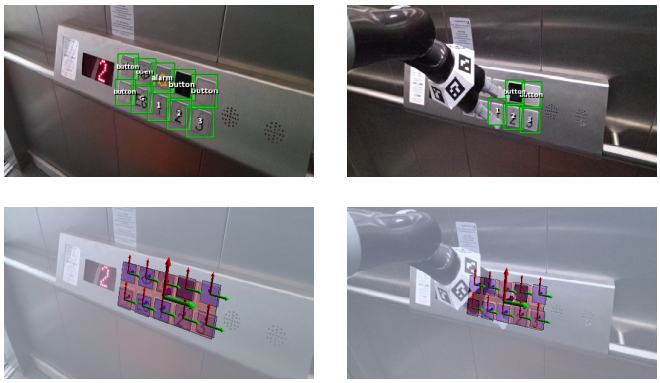

Fig. 5: Examples of exact button panel localization in case of occlusion (right) and in normal conditions (left). top: raw detections from the Faster RCNN; bottom: estimated 6D pose of button panels and respective buttons

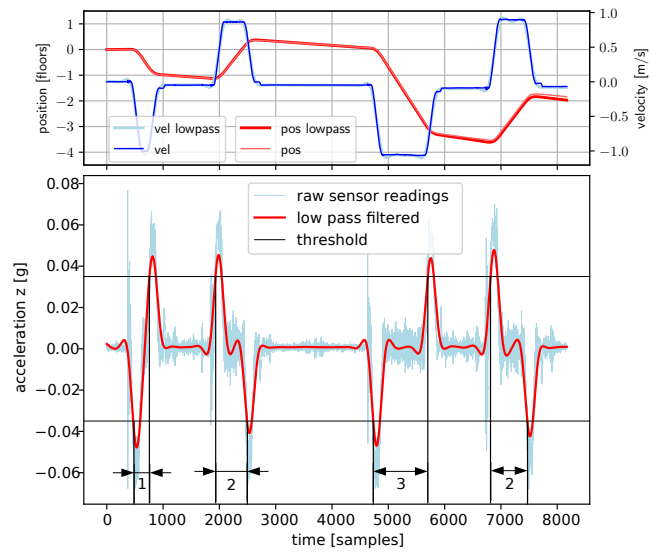

Fig. 6: Bottom: vertical acceleration sensor readings while traveling one floor up, two floors down, three up, two down. The time intervals between the acceleration peaks indicate the floor difference. Top: integrated acceleration (velocity) and integrated velocity (position) show a drift (red curve should end at start level).

not match and 0 otherwise. This ensures that geometrically ambiguous arrangements of buttons can be resolved by means of the associated labels. See Sec. VIII for selection of ρ. The parameter σ has been set to 0.03 m, which defines the matching radius to the order of a single button's size.

The pairing of detection and panel button with the maximum matching score yields the final hypothesis for the panel's pose estimation.

Fig. 5 shows some examples of the detection, which is robust even if more than half of the panel is occluded by the robot's gripper.

VI. Elevator Analysis

Once the lift call button has been pressed, the robot needs to detect when the cabin is ready for boarding. Therefore, the laser range scan is analyzed, while the robot is waiting in front of the door. Individual scan rays' line segments in map coordinates are intersected with a virtual line 5 cm behind and parallel to the door. Then the left most and right most intersection point on that door line define the traversable gap. A threshold on the gap's width allows to distinguish door open and door closed state.

This method also works from inside the cabin, but in this case the virtual line is offset 5 cm to the outside of the door in order to become tolerant against localization errors.

Once the door is open, the robot needs to decide whether the cabin is empty or occupied. Due to safety reasons, we do not allow the robot to enter an occupied cabin, as people could be blocked to leave the lift. To check the cabin, again the laser range scan can be used. If the number of scan end points inside the polygon of the cabin exceeds a threshold, the cabin is considered to be occupied. To compensate for small localization deviation the polygon has a safety margin of 10cm to the actual walls of the cabin. Also people in front of the elevator need to be recognized when the robot has to decide whether to wait outside the lift or when leaving the cabin. To that end a square of 1.5m by 1.5m in front of the elevator's door has been defined as the waiting area. This area can be checked in the laser range scan similar to the inside of the cabin. Usually, there are no other dynamic obstacles in that region other than people.

VII. Floor Recognition

After the robot finally has entered the cabin and the destination floor has been selected by means of another button action, the system needs to detect the current floor in order to leave the cabin at the right one.

While other implementations try to read the indicator screen inside the cabin [11], we rely on the built in acceleration sensor of the SCITOS G5 robot.

We found that simply integrating the vertical component of the acceleration two times is not reliable enough. Even after careful calibration, the resulting distances drift rapidly over more than the height of one story. Fig. 6 (top) shows that drift in the red curve, which in practice should end at the same level as it started. Instead, we found that in most elevators the duration for changing from one specific floor to each other are more or less constant (see Fig. 6 (bottom)). This is especially true, if the weight of the cabin is constant which is guaranteed since the robot will ride the elevator alone. Therefore, by using the acceleration sensor, we simply detect the acceleration event of the cabin at the start and the deceleration event at the end by a simple threshold on the low-pass-filtered vertical acceleration value. The low pass filter shown in Fig. 6 ensures that little bumps at the entrance of the lift do not trigger floor change recognition when the robot enters or leaves the lift. This approach may reach its limits, when the number of stories increases. The potential deviation of the travel time increases with the overall distance

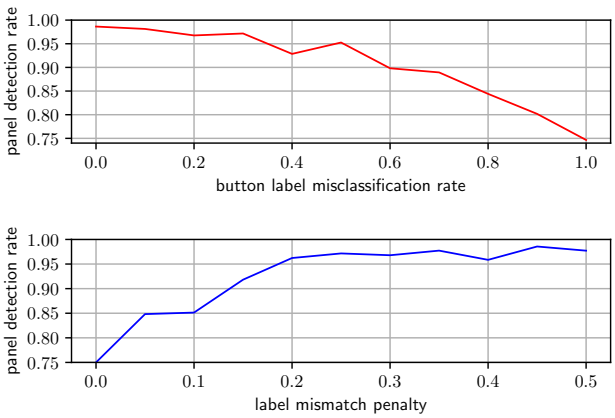

Fig. 7: Ratio of correct panel localization depending on the amount of misclassifications of the button labels (top) and in dependency of the label mismatch penalty parameter ρ (bottom). Both were evaluated over a sequence with heavy occlusions of two button arrays.

and finally reaches the duration of a single floor transition with an increasing number of stories. In our test facilities we only had access to a four story elevator. Therefore, during our real world application the floor recognition did not fail a single time.

VIII. Experimental Evaluation

We use the proposed methods in two installations of the autonomous assistance robot. One is operating in our university building and the other in the target facility, a nursing home for elderly in Ilmenau Germany. First, the results of a quantitative evaluation of the button panel localization is presented before the insights into the development process and the results of a one week application test are discussed.

A. Evaluation of the Button Panel Localization

In the following, we report the analysis of the button panel localization which is a prerequisite for the precise robot localization and button interaction.

In order to evaluate its robustness, we have recorded a video sequence of the robot observing the button panels of two different elevators. This dataset shows occlusions to parts of the panel due to the robot's hand operating on the buttons in 35% of the images.

First, we wanted to show the panel detection results in presence of misclassified buttons in the raw detections. Unknown buttons will effect the panel localization in the same way. This would be the case in a building with more than 10 stories since we only use the 16 button classes as described in Sec. V-A. For the experiment, the predicted labels in the video sequence have been artificially invalidated randomly for a variable amount of raw detections in each image. The label match parameter ρ has been set to 1.0 for that experiment. Fig. 7 (top) shows that for the given setup the panels have been localized correctly in more than 97% of the frames with label drop out rates of up to 30%. The position of the detected panels has been count as false if it snapped to another grid position, which means that there was a position offset of more than 5 cm in 3D world coordinates. Note, that this is not the accuracy of the localization in camera coordinates since it includes the localization error of the moving robot platform.

The position accuracy of the correctly detected panels in camera coordinates inherits directly from the accuracy of the raw button detection and has been evaluated as follows. Knowing the size s of the buttons in the real world, the position accuracy could be evaluated by comparing the offset d of the detected box to the ground truth box and the ground truth box dimensions w.

$$\bar{e} = \frac{1}{N}\sum_{n=1}^{N} s_n \frac{d_n}{w_n} \qquad \hat{e} = \max_{n\in[1,N]} s_n \frac{d_n}{w_n} \tag{2}$$

Using the test dataset of known buttons, the average and max position offset has been found to be $\bar{e} = 0.16\,\text{cm}$ and $\hat{e} = 0.33\,\text{cm}$. This small offset is increased slightly when the 3D position is evaluated from the point cloud data but at the end it is reasonable for hitting the button with the robot's finger.

A further evaluation deals with the free parameter ρ of the panel matching algorithm. Although, it is possible to find the correct matching buttons pairs without any label information if the corners of the panel are not occluded, in cases with the arm in front of the panel the correct pairing based on the label makes the localization of the panel more robust. Fig. 7 (bottom) shows the influence of the parameter ρ. When label matches are not rewarded at all ($\rho = 0$), then due to occlusion 25% of the frames yield a panel detection that is snapped to the wrong grid position. When ρ reaches 0.2, the association mistakes can be reduced and the false panel detections drop to about 3%. The error that might be introduced by a ρ that is too high depends on the actual misclassification rate of the raw button detector. For our practical application that rate is so low that no negative effects of a big ρ could be observed.

B. Application Test and General Findings

The development of the elevator procedure took place in the said facilities leading to a functional transportation service that has been tested in our office building for a week. There were 96 transportation tasks that required changing the floor. In three cases a human intervention was necessary to recover the robot from a deadlock. In one case the robot got stuck when leaving the cabin. The drive system was not able to overcome the gap. In the other cases the robot stopped assuming to be in a collision with the static environment, which actually was not the case. This can happen when the platform gets moved passively or the localization system changes the position without an actual movement of the robot. In the later case the robot model virtually is placed inside e.g. a wall, which causes the collisions.

During the application no problems with the floor recognition could be recorded. The travel-time-based tracking of floors seems to be robust against all jerks happening.

There was a total number of 305 press button procedures, of which seven missed to hit a button at all. That means the force threshold has not been exceeded, which could be caused by a unfavorable positioning of the robot in relation to the button panel of a wrong localization of the panel (distance estimated too large). The unexpected high number of button operations is caused by repeatedly calling the elevator when it is occupied or busy. The timeout for a retry was only 20 sec. When the lift is stopping at other floors this often is not long enough.

During development, the robot occasionally pushed away itself while pressing the call button in front of the lift. Then the changed orientation caused difficulties with the following detection and cabin entering movements. This misbehavior could be mitigated by means of active breaking when the push trajectory is executed with the arm.

A further, more critical point of failure was related to the navigation while entering the cabin. When the robot struggles to get over the doorstep it can get stuck when the cabin doors close and hit the robots bumper. In that case the hardware needs 3 seconds until safety stop is released and the door closes again soon. A fallback strategy for this situation has been implemented, which consists of a manual drive command for going backwards for 10 cm in order to escape from that loop.

IX. CONCLUSIONS

In this paper we present a complete pipeline for automated operation of an elevator by a mobile robot. We introduce our button localization approach based on the complete panel, which makes it robust to detection errors made by the button detector. Additionally, by localizing the panel our approach can handle occluded buttons during push operation allowing us to continuously track the button's position.

We evaluated the panel localization and the complete pipeline through real world trials with two different elevators. While most of the problems remaining are caused by strict time constraints caused by the small time frame in which the elevator doors are open, our pipeline is effective enough to be used in real world applications.

REFERENCES

[1] J.-G. Kang, S.-Y. An, and S.-Y. Oh, "Navigation strategy for the service robot in the elevator environment," in *2007 International Conference on Control, Automation and Systems*. IEEE, 2007, pp. 1092–1097.

[2] H.-M. Gross, A. Scheidig, K. Debes, E. Einhorn, M. Eisenbach, S. Mueller, T. Schmiedel, T. Q. Trinh, C. Weinrich, T. Wengefeld, *et al.*, "Roreas: robot coach for walking and orientation training in clinical post-stroke rehabilitation—prototype implementation and evaluation in field trials," *Autonomous Robots*, vol. 41, pp. 679–698, 2017.

[3] J. Liebner, A. Scheidig, and H.-M. Gross, "Now i need help! passing doors and using elevators as an assistance requiring robot," in *Social Robotics: 11th International Conference, ICSR 2019, Madrid, Spain, November 26–29, 2019, Proceedings*. Springer, 2019, pp. 527–537.

[4] S. Rosenthal and M. Veloso, "Mobile robot planning to seek help with spatially-situated tasks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 26, no. 1, 2012, pp. 2067–2073.

[5] F. Babel, P. Hock, J. Kraus, and M. Baumann, "Human-robot conflict resolution at an elevator - the effect of robot type, request politeness and modality," in *2022 17th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2022, pp. 693–697.

[6] W.-t. Law, K.-s. Li, K.-w. Fan, T. Mo, and C.-k. Poon, "Friendly elevator co-rider: An hri approach for robot-elevator interaction," in *2022 17th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 2022, pp. 865–869.

[7] D. Troniak, J. Sattar, A. Gupta, J. J. Little, W. Chan, E. Calisgan, E. Croft, and M. Van der Loos, "Charlie rides the elevator – integrating vision, navigation and manipulation towards multi-floor robot locomotion," in *2013 International Conference on Computer and Robot Vision*, 2013, pp. 1–8.

[8] D. Zhu, T. Li, D. Ho, T. Zhou, and M. Q. Meng, "A novel ocr-rcnn for elevator button recognition," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3626–3631.

[9] P.-Y. Yang, T.-H. Chang, Y.-H. Chang, and B.-F. Wu, "Intelligent mobile robot controller design for hotel room service with deep learning arm-based elevator manipulator," in *2018 International Conference on System Science and Engineering (ICSSE)*. IEEE, 2018, pp. 1–6.

[10] 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," in *2016 IEEE 20th Jubilee International Conference on Intelligent Engineering Systems (INES)*. IEEE, 2016, pp. 45–50.

[11] J. Lee, X. Cui, H. Kim, S. Lee, and H. Kim, "Elevator riding of mobile robot using sensor fusion," in *The 8th International Conference on Robotic, Vision, Signal Processing & Power Applications: Innovation Excellence Towards Humanistic Technology*. Springer, 2014, pp. 89–98.

[12] R. Stricker, S. Müller, E. Einhorn, C. Schröter, M. Volkhardt, K. Debes, and H.-M. Gross, "Interactive mobile robots guiding visitors in a university building," in *2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication*. IEEE, 2012, pp. 695–700.

[13] J. Krejsa, S. Vechet, K.-S. Chen, M. Havelka, and M. Černil, "Mobile robot in the elevator: What floor am i on?" in *2022 20th International Conference on Mechatronics-Mechatronika (ME)*. IEEE, 2022, pp. 1–5.

[14] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust monte carlo localization for mobile robots," *Artificial intelligence*, vol. 128, no. 1-2, pp. 99–141, 2001.

[15] S. Müller, T. Q. Trinh, and H.-M. Gross, "Local real-time motion planning using evolutionary optimization," in *Towards Autonomous Robotic Systems: 18th Annual Conference, TAROS 2017, Guildford, UK, July 19–21, 2017, Proceedings 18*. Springer, 2017, pp. 211–221.

[16] St. Mueller, B. Stephan, and H.-M. Gross, "Mdp-based motion planning for grasping in dynamic szenarios," in *Europ. Conf. on Mobile Robotics (ECMR), Bonn, Germany*. IEEE, 2021, p. 8 pages.

[17] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320314000235