Intelligent Mobile Robot Controller Design for Hotel Room Service with Deep learning Arm-Based Elevator Manipulator

Po-Yu Yang, Tzu-Hsuan Chang, Yu-Hao Chang, Bing-Fei Wu, IEEE Fellow

***Abstract*— Although mobile robots have achieved great success in indoor navigation, they are still facing problems like operating through multi-floor hotel buildings. In this paper, a low-cost and lightweight mobile robot is proposed for hotel room service. For delivering any item that customers need, the robot should be able to manipulate elevators in an unmanned hotels. So, it is separated into three parts: elevator button detection, coordinate transform and fuzzy logical control to manipulate the robotic arm. To recognize and detect a variety of elevator panels and buttons, a large number of the images are collected and labeled manually by ourselves. With state-of-the-art deep learning framework, our model has achieved 95.172 mean average precision (mAP) even in elevators that are not included in training data. After properly detecting the elevator buttons, the 3D position corresponding to each elevator button is transformed by the fusion of two sensors which are a mono-camera and a Lidar. This paper presents a coordinates transform neural network to estimate real world position, and our method achieves in an average distance error of 1.356 mm. The fuzzy logical controllers are designed for manipulating the robotic arm fast and smoothly.**

Keywords— Deep learning, object detection, sensors fusion, robotic arm manipulating, fuzzy logical control.

I. INTRODUCTION

Service robots have become more and more common in these years. They are applied to carry out the transportation and the guiding mission in the various environment like hotels [1], medical centers [2] and shopping malls [3]. In order to transport to different floors, the robot has to call the elevator, find the panel, select the button of the destination floor, detect the current floor, and confirm the status of the elevator door.

In related researches, a multiple partial models method was used to detect the panels. Hough transform handled the detection of the button edges and the button was recognized by multiple symbols [4]. Yu, et al. applied two ultrasonic distance measure devices to operate the elevator [5]. Wang, et al. proposed line pattern on the elevator panels for finding easily. By doing binary and comparing the characters, the robot could recognize the buttons. There are a camera and a touch sensor set on the tip of the robot arm. Then, it used inverse kinematics to control and coordinate transformation to press buttons [6]. Klingbeil, et al. combined EM algorithm, standard sliding-window object detector, OCR techniques, and HMM to detect the elevator panels and buttons. For the part of character recognition, it used LeNet-5 convolutional neural network. The vision module with a 3D sensor and two cameras estimated the transformation between scene and image points [7]. Dong et al. presented a method for object detection and elevator button recognition by convolutional neural networks (CNN) base. They obtained the contours of the button through edge detection [8]. In other method of detecting elevator buttons, Troniak, et al. used feature matching to detect button and 3D camera to find buttons locations [9]. Islam, et al. combined color thresholding, noise elimination, and button number extraction to overcome special cases like elevator with glass windows. Some properties such as transparent material or reflective effect make detection more difficult [10].

In this paper, we collect and label a large number of the images of elevator panels and buttons. Compared to previous work focusing on single elevator, our model is able to detect different types or shapes elevator panels and buttons. The elevator button detection model is training in our dataset with deep learning and imposing some restraints for using in detecting buttons. Next, we discuss about camera and Lidar fusion. Conventional methods like least square criterion, backward projection, and learning methods are used to estimate 3D coordinate from these two sensors, while they are not enough to estimate properly. Accordingly, we presents a combination of conventional methods and neural network to provide smaller errors. By the help of deep learning methods mentioned above, our robot system is able to achieve such great performance with lightweight and low-cost sensors.

II. SYSTEM STRUCTURES

The entire robotic platform is shown in Figure 1. There are four MECANUM wheels and MD36 DC motors in the mobile platform for omnidirectional movement. A flash micro-controller which is called STM32 sends command to the motors by connecting BTN7971 driver through serial ports and the controller is PID. The maximum speed of the motors is 2 m/sec. The Lidar of the type RPLIDAR A1 is used for SLAM, obstacle avoidance, and distance estimation.

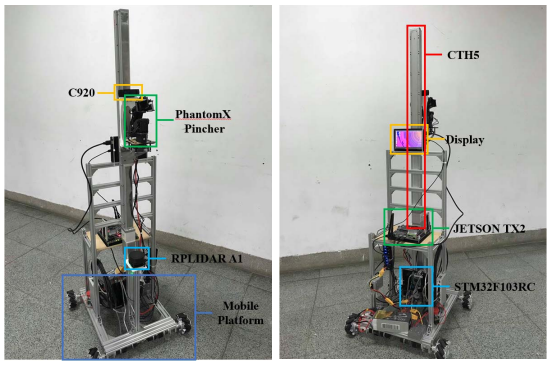


Figure 1. The robotic platform with devices mounted.

*Resrach is supported by MOST 105-2221-E-009-041.

For operating the elevator autonomously, a slide rails which is of the type TOYO CTH5 and a 4-joint robotic arm which is of the type PHANTOMX Pincher are mounted on the platform. The robotic arm can press elevator buttons of various heights by moving the slide rails vertically. The length of the slide rails is 80 cm and the maximum speed of it is 50 cm/sec. The robot can process elevator button detection and image coordinates transformation on the robotic arm by a mono-camera. The camera mounted on the robot is a low-cost Logitech C920. For taking a clear look, the system structure and the system flow chart are shown in Figure 2.

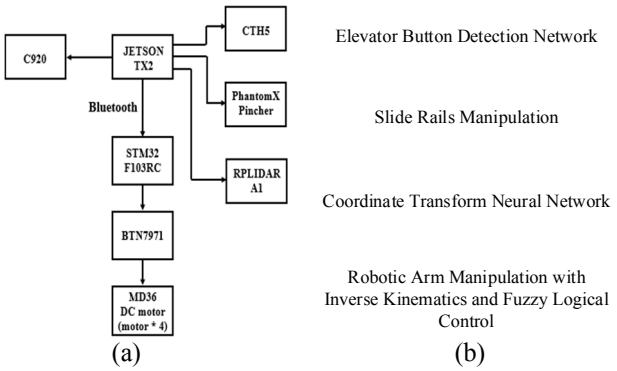


Figure 2. (a) The system structure. (b) The system flow chart.

In order to solve the radial distortion and tangential distortion problems of the camera, we need to do the calibration first. There are two software packages such OpenCV [11] and MATLAB calibration toolbox [12].

The control center is NVIDIA JETSON TX2 which is an AI supercomputer on a module. It is responsible for sending the commands to all devices and computing deep learning algorithm. Besides, the communication between the TX2 and micro-controller STM32 is via Bluetooth. Power is provided by the 22.2V/ 16Ah Li-Po battery. The battery outputs 24V to the slide rails through the regulator circuit. Also, it outputs 12V to the arm and 19V to TX2 through the buck circuit. The Lidar and the camera is powered by USB of TX2.

III. Elevator Buttons Detection

To find the target button, the robot must localize the buttons on the panel and recognize the symbol of buttons. Therefore, a buttons detection model is proposed.

The images of elevator panels and buttons are required by the model, but there is no open dataset on the internet, so the dataset is captured by ourselves. Currently, there are 260,560 images which are composed of 8 different kinds of elevator panels (see Figure 3.). The dataset contains elevator inside and outside panels and images in the dataset are captured with different shot angles. In addition, all the panels and buttons are marked and labeled in the images manually. The labels are divided into 26 classes, including inside and outside panels, buttons about upstairs, downstairs, floor numbers from B1 to 9, and their corresponding bright patterns.

To avoid insufficient images for training, the data are expanded by doing histogram equalization, blurring, and sharpening (see Figure 4.). It can also offset the random illumination and camera shake rendered in the images. There are 1,042,240 images after data expansion. They are separated into two parts (1,011,570 images for training and 30,670 images for validation).

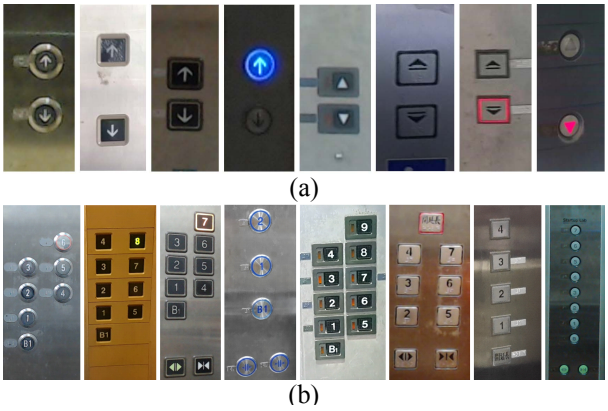


Figure 3. Elevator panels in our dataset. (a) Inside buttons. (b) Outside buttons.

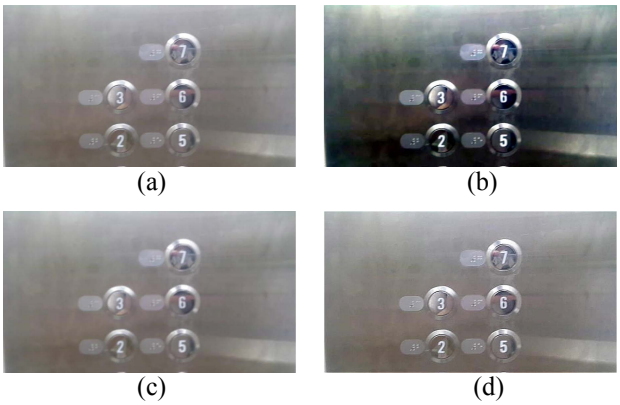


Figure 4. Data expansion. (a) Original image. (b) Histogram equalization. (c) Blurring. (d) Sharping.

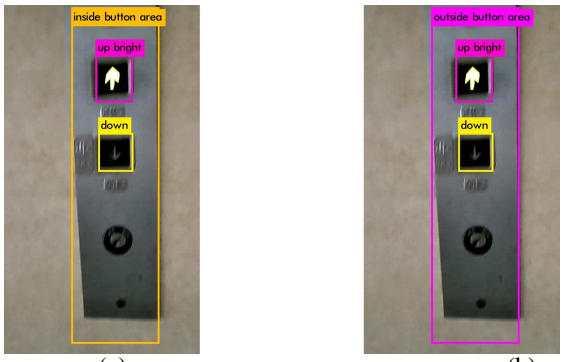
Due to speed of operation, YOLO v2 [13] is chosen and their convolutional layers are pre-trained on our dataset to obtain a yolo-based elevator buttons detection model. However, the result of the model comes about label overlapping or misidentification occasionally. So, we impose restraints on the result. For label overlapping, a panel cannot have more than one button in same patterns. The label which has the highest confidence is selected consequently (see Figure 5.).

Besides, inside or outside panel is distinguished from the quantities of inside labels and outside labels to reduce mistakes. If the number of outside labels is more than inside labels, the outside panel is determined and all the inside labels are filtered out (see Figure 6.).

With this detection model, the robot can detect, position, and classify the buttons in the frames correctly. Moreover, we apply the model to label other elevator images for getting more data.

(a) (b)

Figure 5. (a) Label overlapping. (b) Adding restraints to resolve label overlapping.



(a) (b)

Figure 6. (a) Label misidentification. (b) Adding restraints to resolve label misidentification

IV. Mono Camera and Lidar Fusion

In our experiments, mono camera and 2D Lidar equipped on our robot will be able to estimate 3D coordinate in order to tell robotic arm where the elevator buttons are located. To estimate the elevator buttons' position, simple assumption that elevator panels are perpendicular to the ground is proposed. Consequently, the raw 2D Lidar data are able to determine a 3D plane by least square criterion. And, the backward projection is proposed to transform 2D points from camera coordinate onto 3D plane which is estimated from 2D Lidar. The camera intrinsic and extrinsic parameters being used in backward projection process are measured by mono camera calibration. The calibration process uses a MATLAB software package.

A. Apply Least Square Criterion to Approach a Plane

The RPLIDAR sensor is a 360 degrees Lidar, and the front 90 points, 90 degrees, of Lidar are used to calculate the linear equation. The Lidar data lie on the x-z plane which is parallel to the ground. It could be transferred from distances and angles of each Lidar point as follows:

$$\begin{cases} x = \ell \cos\theta + d_x \\ z = \ell \sin\theta + d_z \end{cases}, \tag{1}$$

where d_x and d_z notes the offset distance from Lidar coordinate to arm coordinate at each axis, ℓ indicates the distance of each point, and θ represents the angle of each point. Then, the plane equation that describes the elevator panels is shown below:

$$\alpha x + \beta y + \gamma z + \delta = 0, \tag{2}$$

$$z = \alpha x + \delta, \tag{3}$$

where β is set to be zero and γ is set to be -1, as there is an assumption that elevator panels are perpendicular to the ground. Due to the same reason, least square criterion is applied to approach a 3D plane as follows:

$$\begin{cases} \alpha = \dfrac{\sum (x-\overline{x})(z-\overline{z})}{\sum (x-\overline{x})^2} = \dfrac{\sum xz - \dfrac{\sum x \sum z}{n}}{\sum x^2 - \dfrac{(\sum x)^2}{n}} \\ \delta = \overline{z} - \alpha \cdot \overline{x} = \dfrac{\sum z - \alpha \cdot \sum x}{n} \end{cases}. \tag{4}$$

As a result, a 3D plane is estimated by 90 points of Lidar data. In order to make camera coordinate transfer to world coordinate, the backward projection is proposed to project points of camera coordinate onto the 3D plane.

B. Backward Projection from 2D points to 3D

To transform camera coordinate back to world coordinate, the backward projection is presented using both intrinsic and extrinsic matrixes of cameras. Forward projection is given by

$$w \odot \vec{\boldsymbol{p}} = \boldsymbol{K}\left(\boldsymbol{R}\vec{\boldsymbol{P}} + \vec{\boldsymbol{t}}\right), \tag{5}$$

where $\boldsymbol{K}_{3\times3}$ indicates the intrinsic matrix, $\boldsymbol{R}_{3\times3}$ notes the rotation matrix, and $\vec{\boldsymbol{t}}_{3\times1}$ presents the translation vector. Moreover, $\vec{\boldsymbol{P}}_{3\times1} = [U, V, W]^T$ is a point on world coordinate, $\vec{\boldsymbol{p}}_{3\times1} = [u, v, 1]^T$ is a point on camera coordinate, and w is a scaling factor. $\odot$ is the element-wise product.

$$\vec{\boldsymbol{P}} = w \odot \boldsymbol{R}^{-1}\boldsymbol{K}^{-1}\vec{\boldsymbol{p}} - \boldsymbol{R}^{-1}\vec{\boldsymbol{t}}. \tag{6}$$

Points from elevator panels have to lie on the plane being estimated previously, so the points have to satisfy the equation below, where $\vec{\boldsymbol{n}} = [\alpha, \beta, \gamma]$ is the normal vector of the plane.

$$\vec{\boldsymbol{n}}\vec{\boldsymbol{P}} + \delta = 0. \tag{7}$$

$$\vec{\boldsymbol{n}}\left(w \odot \boldsymbol{R}^{-1}\boldsymbol{K}^{-1}\vec{\boldsymbol{p}} - \boldsymbol{R}^{-1}\vec{\boldsymbol{t}}\right) + \delta = 0. \tag{8}$$

$$w = \frac{\vec{\boldsymbol{n}}\boldsymbol{R}^{-1}\vec{\boldsymbol{t}} - \delta}{\vec{\boldsymbol{n}}\boldsymbol{R}^{-1}\boldsymbol{K}^{-1}\vec{\boldsymbol{p}}}. \tag{9}$$

As the scaling factor is determined, it is substituted for the equation (6) to get real position on world coordinate.

In addition, it is also possible to calculate multiple points at a time by expanding the vector $\vec{p}_{3\times1}$ to matrix, while some of the operators have to change. The original division of scaling factors is going to do element-wise division, and original product indicates element-wise product.

C. Coordinate Transform Neural Network

There are some risks that the linear approach method has its own limitation to estimate coordinate properly if walls are not purely straight. Therefore, based on previous results, a neural network (TABLE I.) is trained to reduce the uncertainty and overcome this weakness. The training data which include raw RPLIDAR point clouds, image coordinate and algorithm-based world coordinate are collected.

TABLE I. COORDINATE TRANSFORM NETWORK

Layer		*Dimension*	*Channel*	*Description*
Input_1		90	1	Lidar
1d_Conv_1		90	512	Filter: 512
				Kernel Size: 7
				Activation: ReLU
1d_Conv_2		90	256	Filter: 256
				Kernel Size: 5
				Activation: ReLU
1d_Conv_3		90	128	Filter: 128
				Kernel Size: 3
				Activation: ReLU
1d_Conv_4		90	64	Filter: 64
				Kernel Size: 3
				Activation: ReLU
1d_Conv_5		90	32	Filter: 32
				Kernel Size: 3
				Activation: ReLU
1d_Conv_6		90	32	Filter: 32
				Kernel Size: 3
				Activation: ReLU
Pooling_6		45	32	Stride: 2
1d_Conv_7		45	16	Filter: 16
				Kernel Size: 3
				Activation: ReLU
Pooling_7		22	16	Stride: 2
Concat	Flatten	352	1	
	Input_2	2	1	Image Coordinates
	Input_3	3	1	Estimated Coordinates $\vec{P}_{3\times1}$
FC_8		64	1	Activation: ReLU
FC_9		3	1	Activation: Linear
Output_layer		3	1	World Coordinates

Since input data of Lidar sensor are 1-dimensional with 90 features, 1-d convolutional layers are applied. In order to extract features of slope, the kernel size of the first convolutional layer is set to be larger, e.g. 7. And, max pooling layers are reduced for keeping more information, so only two max pooling layers are used. As it is a regression problem with unbounded output values which are positive and negative, the activation function of the last layer is chosen to be linear. The others are set to be ReLU.

V. ARM MANIPULATORS

The robotic arm mounted on the slide rails is PhantomX Pincher which consists of four joints and a gripper. To manipulate this arm, the library of PhantomX Pincher [14] is applied, such as computing forward and inverse kinematics. It is accurate to calculate all kinematics parameters. But, due to its own limitation of servo motors (Dynamixel AX-12) which have lower torques, some errors between desired and real end-effector's position should be well controlled.

Fuzzy logical controllers are used to solve these problems. Three fuzzy logical controllers are for x-axis, y-axis, z-axis errors, separately. X-axis is for forward and backward (FB), y-axis is for left and right (LR), and z-axis is for up and down (UD). The fuzzy table of the forward-backward output is in TABLE II. The height of the end-effector is more difficult to control than the left-right position, so the input of this controller is up-down and forward-backward position. If the end-effector is far from the destination (positive medium or positive small), the controller would make it go slower forward to modify its direction. If the end-effector is too close to the target position (negative medium or negative small), the arm would stop immediately or even go backward. The special case occurs when both forward-backward and up-down inputs are all zero, i.e. the end-effector is absolutely at the right position. We consider the pressing process against the elevator buttons, so the controller assigns a larger force (positive medium) to make sure the arm has enough strength.

TABLE II. FORWARD-BACKWARD FUZZY TABLE

FB / *UD*	*NM*	*NS*	*Z*	*PS*	*PM*
NM	*NM*	*NS*	*Z*	*Z*	*PS*
NS	*NS*	*Z*	*Z*	*Z*	*PS*
Z	*Z*	*Z*	*PM*	*PS*	*PM*
PS	*NS*	*Z*	*Z*	*Z*	*PS*
PM	*NM*	*NS*	*Z*	*Z*	*PS*

VI. EXPERIMENTAL RESULTS

The software of our robot system is developed on Robot Operating System [15], Kinetic version, which is compatible for both C/C++ and Python. Python is applied in order to implement TensorFlow [16] and Keras [17] frameworks for CNN-based coordinate transform. And, the rest of our programs such as the detection and the arm manipulator run in C/C++.

A. Elevator Buttons Detection

The elevator buttons detection model is applied to label the location of target buttons. For assessing the effect of the

model, the testing images of the elevator which is not included in the training data are shown in Figure 7. Total 991 images as testing data. TABLE III. shows the performance of the detection models. Compared to the yolo-based elevator buttons detection model getting 64.827 mAP, our detection model achieves 95.172 mAP by applying specific restraints mentioned in section III.

(a) (b)

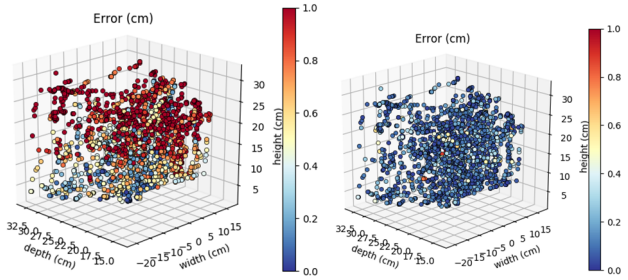
Figure 7. (a) The inside panel of the elevator for testing. (b) The outside panel of the elevator for testing.

TABLE III. THE PERFORMANCE OF DETECTION MODEL

Detection Model	Mean Average Precision
Our elevator buttons detection model	95.172
Yolo-based elevator buttons detection model	64.827

B. Algorithm-and-CNN-based Coordinate Transform

The intrinsic parameters are calibrated by MATLAB software first. The translation vector from camera to world coordinate is measured manually, while the rotation matrix from these two coordinates is assumed to be an identity matrix. To evaluate and verify results of the coordinate transform method, a checkerboard's corners detection function based on OpenCV is used to determine the ground truth of world coordinate. After applying the algorithm-based coordinate transform, average distance errors of each pixel on the image coordinate are shown in Figure 8.



(a) (b)

Figure 8. Average distance errors of each pixel point. (a) Algorithm-based method on world coordinate. (b) Algorithm-and-CNN-based method on world coordinate.

This method results in an average error of 7.12 mm. As you can see, larger errors occur on upper half of entire workspace because of the small deviation of extrinsic parameters between camera, RPLIDAR and world coordinate. That is hard to observe when measuring manually.

In our experiment, a neural network model is proposed to calibrate this type of errors. For collecting the training data, the total numbers of them are about 10,263 samples which are separated into three parts. For the number of training, validation and testing data, there are 7,389, 821 and 2,053 samples, respectively. At the training stage, the batch size is 64, the learning rate is 0.0005, and the loss function is mean absolute error (MAE). After 5,000 epochs, the model results in an average error of 1.356 mm on testing data shown in TABLE IV. The previous work [7] and the joint calibration method [18] apply Levenberg-Marquardt algorithm to converge extrinsic and intrinsic matrixes to the local minima. It is implemented by MATLAB software packages. With the same devices and sensors, i.e. a mono-camera and a Lidar, it is also assumed that the elevator panels are vertical against floors. Compared with previous works which result in an average error of 1.455 mm shown in TABLE IV. our model-based method has more flexibilities and abilities to overcome different types of errors. Although most of the errors are caused by extrinsic and intrinsic parameters, there are still errors from Lidar sensor and the linear models of the least square criterion. As elevator panels are not always to be quite straight or continuous, our method with both algorithms and neural networks is said to be more strength to deal with those problems.

TABLE IV. COORDINATE TRANFORM PERFORMANCES

Method	Average Error
Klingbeil, et al. [7]	1.455 mm
Our method	1.356 mm

a. The average errors are the Euclidean distance errors

C. Arm-Based Elevator Manipulator Results

The robotic platform was initially placed toward the elevator panel within a proper distance that the manipulator can stretch to touch the buttons, and was demanded to press the 'down' outside button as shown in Figure 9. It was also commanded to press the '1F' inside button as shown in Figure 10. .

First, the slide rail started to go upward from the bottom. As the target button was detected by the elevator button detection model, the slide rail stopped. The slide rail took about 5 seconds to complete which depends on how high the target button was. The speed of the slide rail is set to be 10 cm/s which is not quite fast to avoid getting blurred frames. Since the slide rail stopped, the coordinate transform network was applied to get the world coordinate, and it took about 100 milliseconds to complete. The final state was robotic arm manipulation with fuzzy logical control. It took 5 seconds in the pressing process, and took another 5 seconds leaving the button area. See *https://youtu.be/f6J3bLgF554* for a video showing clips from the experiments.

Figure 9. The outside button (down button) is pressed by the robotic arm

Figure 10. The inside buttons (1F button) is pressed by the robotic arm.

VII. CONCLUSION

This paper has implemented an intelligent mobile robot which can operate elevator autonomously with deep learning models in detecting buttons and producing 3D coordinates. The elevator buttons detection model is trained with a large number of data for elevator panels and buttons which are collected and labeled by ourselves. The model is improved by imposing some restraints for reducing the mistakes during detecting. Testing in an unknown elevator, our detection model can also find target buttons in high accuracy. Moreover, the model of estimating 3D coordinates overcomes noises from low-cost sensors. The result of the model can come out in small distance errors for the robot manipulator to press buttons properly and successfully.

ACKNOWLEDGMENT

This work is supported by the Ministry of Science and Technology under Grand no. MOST 105-2221-E-009-041.

REFERENCES

[1] H. Osawa *et al.*, "Analysis of robot hotel: Reconstruction of works with robots," in *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2017, pp. 219-223.

[2] A. Vasquez, M. Kollmitz, A. Eitel, and W. Burgard, "Deep Detection of People and their Mobility Aids for a Hospital Robot," in *2017 European Conference on Mobile Robots (ECMR)*, 2017, pp. 1-7.

[3] R. Gillani and A. Nasir, "Incorporating artificial intelligence in shopping assistance robot using Markov Decision Process," in *2016 International Conference on Intelligent Systems Engineering (ICISE)*, 2016, pp. 94-99.

[4] P. Mukhopadhyay and B. B. Chaudhuri, "A survey of Hough Transform," *Pattern Recogn.,* vol. 48, no. 3, pp. 993-1010, 2015.

[5] X. Yu, D. Li, L. Liyuan, and H. Kah Eng, "Lift-button detection and recognition for service robot in buildings," in *2009 16th IEEE International Conference on Image Processing (ICIP)*, 2009, pp. 313-316.

[6] W. J. Wang, C. H. Huang, I. H. Lai, and H. C. Chen, "A robot arm for pushing elevator buttons," in *Proceedings of SICE Annual Conference 2010*, 2010, pp. 1844-1848.

[7] E. Klingbeil, B. Carpenter, O. Russakovsky, and A. Y. Ng, "Autonomous operation of novel elevators for robot navigation," in *2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 751-758.

[8] Z. Dong, D. Zhu, and M. Q. H. Meng, "An autonomous elevator button recognition system based on convolutional neural networks," in *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2017, pp. 2533-2539.

[9] D. Troniak *et al.*, "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.

[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)*, 2016, pp. 45-50.

[11] Itseez. (2015). *Open Source Computer Vision Library*. Available: https://github.com/itseez/opencv

[12] "MATLAB and Statistics Toolbox Release 2017a," ed: The MathWorks, 2010.

[13] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6517-6525.

[14] M. K. (2017, March, 12). *Hardware and support package for the PhantomX Pincher robot arm*. Available: https://github.com/rst-tu-dortmund/pxpincher_ros/tree/kinetic-devel

[15] M. Quigley, Conley, Ken., Gerkey, Brian P.., Faust, Josh., Foote, Tully., Leibs, Jeremy., Wheeler, Rob., and Ng, Andrew Y., "ROS: an open-source Robot Operating System," presented at the ICRA Workshop on Open Source Software, 2009.

[16] A. A. Martín Abadi, Paul Barham, Eugene Brevdo, *et al.* (2015, March, 12). *TensorFlow: Large-scale machine learning on heterogeneous systems*. Available: https://www.tensorflow.org/

[17] F. Chollet. (2015, March, 12). *Keras*. Available: https://github.com/fchollet/keras

[18] Q. V. Le and A. Y. Ng, "Joint calibration of multiple sensors," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 3651-3658.