Autonomous Operation of Novel Elevators for Robot Navigation

Ellen Klingbeil, Blake Carpenter, Olga Russakovsky, Andrew Y. Ng

***Abstract*— Although robot navigation in indoor environments has achieved great success, robots are unable to fully navigate these spaces without the ability to operate elevators, including those which the robot has not seen before. In this paper, we focus on the key challenge of autonomous interaction with an unknown elevator button panel. A number of factors, such as lack of useful 3D features, variety of elevator panel designs, variation in lighting conditions, and small size of elevator buttons, render this goal quite difficult.**

To address the task of detecting, localizing, and labeling the buttons, we use state-of-the-art vision algorithms along with machine learning techniques to take advantage of contextual features. To verify our approach, we collected a dataset of 150 pictures of elevator panels from more than 60 distinct elevators, and performed extensive offline testing. On this very diverse dataset, our algorithm succeeded in correctly localizing and labeling 86.2% of the buttons. Using a mobile robot platform, we then validate our algorithms in experiments where, using only its on-board sensors, the robot autonomously interprets the panel and presses the appropriate button in elevators never seen before by the robot. In a total of 14 trials performed on 3 different elevators, our robot succeeded in localizing the requested button in *all* 14 trials and in pressing it correctly in 13 of the 14 trials.

I. INTRODUCTION

Robots have been able to autonomously navigate unknown building floors for some time; however, their mobility in these general environments is restricted if they are not capable of autonomously operating elevators. Current robot systems (used in environments such as hospitals and labs) either rely on human assistance or use infrared transmitters to interact with an elevator ([1], [2], [3], [4]). Relying on human assistance can be inefficient and one can imagine a situation where it might be impossible, e.g. a robot janitor cleaning a building after working hours. Retrofitting all elevators with infrared detectors could be costly and time-consuming and may not be possible if robots must be able to navigate in a large number of different buildings.

In this paper, we consider the challenge of enabling a mobile robot to autonomously operate elevators (with no human intervention), including those never before seen by the robot. We focus on developing algorithms to enable a robot to identify and accurately localize buttons and recognize their labels, and execute the action of pressing the button corresponding to the desired floor. We specifically address interior button panels since call buttons usually consist of only up/down buttons and represent a constrained special case of general elevator panels.

Ellen Klingbeil is with the Department of Aeronautics and Astronautics, Stanford University. Blake Carpenter, Olga Russakovsky and Andrew Y. Ng are with the Department of Computer Science, Stanford University. {ellenrk,blakec,olga,ang}@cs.stanford.edu

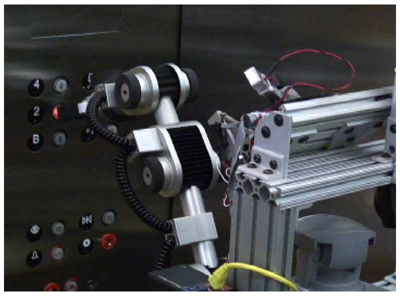


Fig. 1. Robot autonomously operating an elevator

Figure 2 shows some sample elevator panels. Note the wide variability of button and label types and the challenging lighting conditions. To identify the locations and labels of all the buttons in a single 2D image, we use sliding-window object detection and optical character recognition. In Figure 2 again, also note the grid layouts and the sequential (ordered by floor) arrangement of buttons. We use machine learning techniques to take advantage of these contextual cues to improve the performance of the baseline detectors. Specifically, we use Expectation-Maximization to fit the buttons to one or more grids, allowing for recovery of false negatives and reliable label extraction. A Hidden Markov Model is applied to the results from the optical character recognition to correct for mislabeled buttons.

We validate our algorithms in a set of experiments where the robot is commanded to press a given button in an elevator never seen before by the robot. An experimental run is considered a success if the robot locates and presses the appropriate button using only its onboard sensors. The perception algorithm correctly localizes the desired button in all 14 trials and the manipulator presses the desired button in 13 of the 14 trials. On a much more diverse, offline dataset, our algorithm succeeds in correctly localizing and labeling 86.2% of the buttons.

II. RELATED WORK

Several researchers have demonstrated a robot using an elevator *with* human assistance. In the 2002 AAAI Mobile Robot Challenge, one of the subtasks for participating robots required the robot to navigate to a different floor using an elevator. Teams were given pictures of the elevator prior to the challenge, but since the robots were allowed to ask a human for assistance, none attempted to autonomously press the buttons ([1], [2]). Miura et. al developed a framework for interactive teaching of a robot, and used the scenario of elevator operation to test their algorithms. After undergoing a training phase where a human pointed out key information, such as the location of the door and buttons in the elevator,

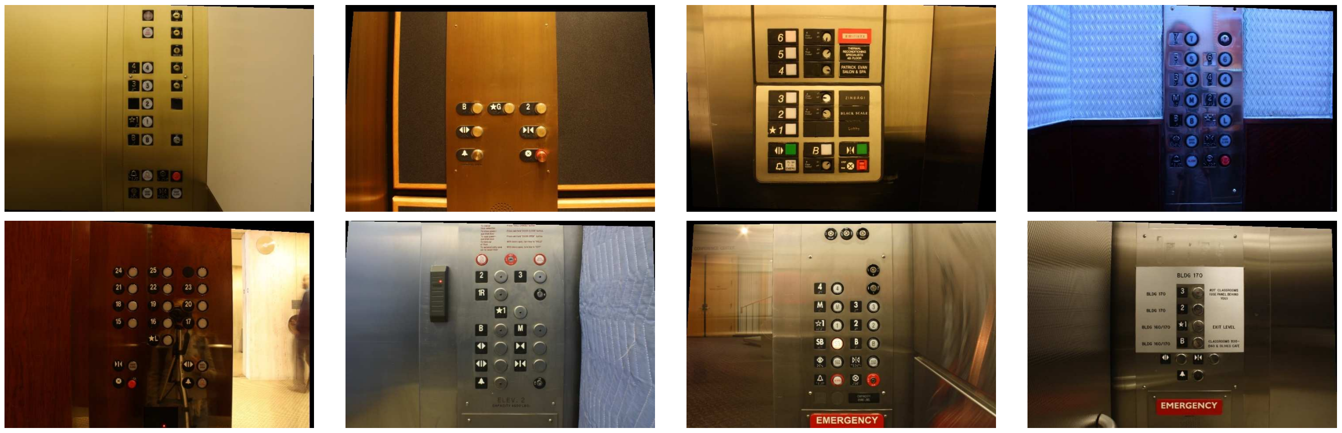


Fig. 2. Sample elevator panels. Notice (1) the wide variability in the button appearance: the shape, the material (metal vs. plastic), the presence or absence of letters and numbers; (2) the imaging conditions in elevators: dim lights, specular reflections; (3) the non-standardized grid arrangement of the buttons; (4) the variety of label types: different fonts, difficult-to-perceive numbers (in some cases rubbed off or damaged); (5) various naming conventions (`L` for lobby versus `G` for the ground floor vs `1`). Our algorithm addresses each of these issues in building a robust automated system for elevator button detection and manipulation.

their robot was able to successfully operate the elevator to traverse a floor of the building [3].

Kang et al. looked at developing a navigation strategy for a robot using a known elevator. They addressed navigation and path planning and also proposed algorithms to recognize the buttons, the direction the elevator is moving, and the current floor for the elevators in a single building. However, since their robot did not have arms, it had to wait for human assistance to press buttons [4]. Recently, there has been some interest in having robots autonomously operate known elevators. In the 2008 TechX Challenge, teams competed towards enabling a robot to autonomously navigate from an outdoor environment to an indoor environment, including operating an elevator to reach a specified floor. The teams were provided with a digital image of the elevator before the competition [5], and the robot was required to operate the single known elevator.

Despite some of these promising results, the difficult challenge of a robot completely autonomously operating unknown elevators is still outstanding. A crucial, unsolved subtask is the development of generalized vision-based algorithms that help the robot localize the elevator buttons and assign appropriate labels with high accuracy. Object detection algorithms based solely on 2D data can suffer from false detections when the objects and/or scenery vary significantly from the data on which the algorithms were trained.[1]

Using contextual cues has shown to improve object detection in 2D images [7], [8], [9]. In the context of an elevator panel, we recognize two main contextual cues: the arrangement of the buttons in one or more grids, and the arrangement of labels in order of floors.

[1]Recently researchers have shown that augmenting 2D images with 3D sensing can provide significant performance boosts to object detection algorithms [6]. However, most current 3D sensors do not have sufficient resolution to detect the buttons at the range required. Thus, in the case of an elevator panel, 3D data provides little information in identifying the location of the buttons other than helping to remove detections that do not lie in the plane of the wall.

Our goal is to enable a robot to completely autonomously operate any elevator in any building, including those that the robot has not seen before. In contrast to many of the recent works on robotic elevator operation, we focus our efforts on developing robust perception algorithms to detect, localize, and label elevator buttons.

III. APPROACH

To successfully operate the elevator, the robot must localize the buttons, recognize the button labels, and control the manipulator to press the button. Following the American Disability Association (ADA) guidelines [10], we make the following assumptions: elevator buttons must be no smaller than 1.9 cm in diameter and button labels must lie to the left of the corresponding button. The button detections must have high precision since buttons may be (and often are) as small as 1.9 cm.

Elevator panels vary widely in appearance and arrangement of buttons. Further, lighting conditions vary greatly among elevator cabs (see Figure 2). Even with a large dataset for training, extracting enough image features which are common across all button variations to be able to recognize buttons in new images (unseen in the training phase) is difficult. Further, correctly labeling the buttons relies on both accurate localization of the label and good performance of the optical character recognition (OCR) algorithm. A straight-forward combination of these two steps is largely inadequate. We develop a more complex model that uses machine learning techniques to incorporate features such as arrangement of the buttons in grid patterns and sequential ordering of labels and demonstrate improvement in overall performance.

First, we use the fact that most elevator buttons lie on a grid to infer missed detections and remove false positives. As shown in Figure 2, the grid stucture varies greatly between panels and thus has to be learned from data for each elevator individually. We apply the Expectation Maximization (EM) algorithm described in detail in section III-B.

Second, we note that elevator buttons generally appear in order of floors: floor 1 is followed by floor 2 and then by floor 3. We can use this knowledge to automatically correct mislabeled buttons: for example, if the optical character recognition (OCR) algorithm labels consecutive buttons `10`, `11`, `11`, `13`, we infer that the second `11` should be changed to a `12`. Hand-coding such rules is difficult: various possible labels exist for the ground floor (`L`, `G`, `1`), unexpected labels often appear (`R`, `S`), and special cases such as the `13th` floor being missing in some buildings have to be taken into account. We automatically learn these rules from training data using a Hidden Markov Model (HMM) as described in section III-D and use it to produce more consistent labels.

Overall, our perception algorithm consists of four main steps: (A) button detection using a standard sliding-window object detector, (B) grid fit using the EM algorithm, (C) label binarization and recognition using OCR techniques, and (D) consistency enforcement using the HMM.

A. Button Detection

We use a 2D sliding window object detector to capture common visual features among elevator buttons. This sliding window object detector, derived from the patch-based classifiers introduced by Torralba [11] and implemented by [12], provides initial estimates for the locations of all the buttons in an image.[2]

Knowledge of average elevator button size (based on ADA guidelines) and the distance of the camera from the panel (obtained from a laser scanner), allows us to compute an upper and lower bound on the expected detection size for each panel. A standard approach for post-processing object detector results is to use a fixed confidence threshold (throw out detections with confidence values less than a fixed value) and then apply non-maximal suppression to remove overlapping detections. However, in our scenario, large variations in imaging conditions among elevators cause the average confidence value for detections to be much lower on some panels than others. Thus, using a fixed confidence threshold results in very few detections on some of the images in our test set (i.e., many false negatives). We instead use a dynamic threshold, determined at run-time for each individual panel, which normalizes the panels, followed by clustering to produce more accurate estimates of button positions (see Figure 3). More details on the button detection post-processing are given in section IV-A.

B. Grid Fit

As shown in Figure 3(c), despite the dynamic thresholding and clustering improvements, the button detection step still

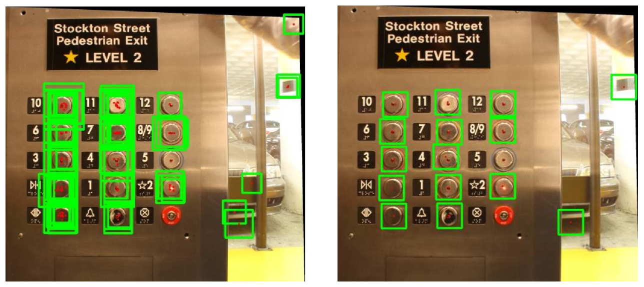


Fig. 3. (a) Output of the sliding window detector. (b) The use of dynamic threshold to remove detections that are of relatively low confidence for each individual panel, followed by clustering.

results in a number of both spurious detections and missed detections. To help eliminate these false positives and recover false negatives, as well as provide an ordering scheme for enforcing label consisteny, we observe that elevator buttons usually lie in one or more grids. This contextual cue is incorporated into our model by fitting the candidates from the sliding window detector to grids using Expectation Maximization (EM) [13].

The EM implementation requires initial estimates for all of the grid parameters. The initialization step clusters all the sliding window detections and estimates a cell width and height by examining clusters along similar horizontal and vertical bands. It then iterates through each cluster and attempts to recursively grow a grid from the current cluster location. This results in a number of possible initial grid fits. We assume that panels have at most five grids, and consider five types of initializations (with one, two, three, four, and five grids respectively), as shown in Figure 4. We compute the log likelihood of all the initial grids (see Equation 3 below), and choose the ones with the maximum likelihood for each of the five types of initialization to input into the EM algorithm.[3]

We use an implementation of EM with a mixture of Gaussians model to learn the best grid parameters. The algorithm is presented in detail in Table I. Briefly, our observations consist of the button detections from the sliding window classifier $\{x^{(1)}, \ldots x^{(m)}\}$. Each button detection $x^{(i)}$ is associated with a hidden variable $z^{(i)}$ assigning it to one of the grid cells. Specifically, if the number of rows and columns in the grid are n_r and n_c respectively, then

$$z^{(i)} \in \{(1,1),(1,2),\ldots,(n_r,n_c),\texttt{outlier}\} \tag{1}$$

We want to model the data by specifying a joint distribution

$$p(x^{(i)}, z^{(i)}) = p(x^{(i)}|z^{(i)}) \times p(z^{(i)}) \tag{2}$$

Table I defines $p(x^{(i)}|z^{(i)})$ formally, but the intuition is as follows: If $z^{(i)}$ corresponds to one of the grid cells then $p(x^{(i)}|z^{(i)})$ follows a normal distribution around the center of the grid cell $z^{(i)}$. Otherwise, $z^{(i)}$ = `outlier` and

[2]Briefly, the supervised training procedure produces a dictionary of features consisting of localized templates from cropped training examples. The relevance of each of these features in identifying whether a button is present in an image patch is evaluated by computing the normalized cross correlation of each template with the image patch. A binary classifier is learned by using boosted decision trees to select the set of feature patches which are most effective at recognizing a button in the training examples (see [11] for more details). We then use this classifier within a sliding-window detector framework to compute the probability of a button within each rectangular subwindow of the elevator panel image.

[3]Since fitting the data to more grids will always produce an increase in the total likelihood, we compare the five grid layouts using a way that penalizes for the increase in the number of model parameters. Specifically, we use the Bayesian Information Criterion (BIC) to choose the best fit: $BIC = -2\ln(L) + k\ln(n)$ where L is the value for the likelihood function, n is the number of button detections, and k is the total number of grid parameters.

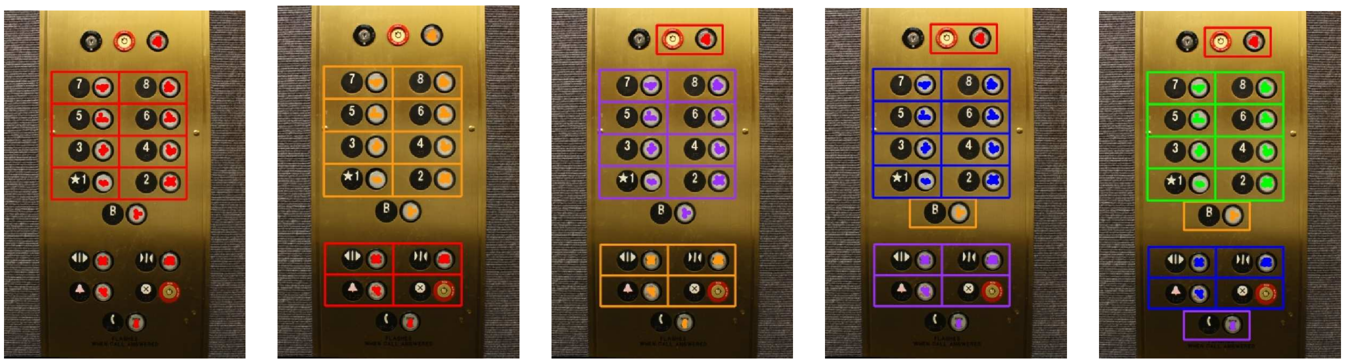


Fig. 4. *Best viewed in color.* Five initializations of grid fit for a single elevator panel, with one, two and three, four, and five grids respectively. These serve as input to the EM algorithm. The colored clusters represent the estimated button centers based on the button detections that fall within each cell.

$p(x^{(i)}|z^{(i)})$ follows a uniform distribution over all positions in the image. We assume a uniform prior for $p(z^{(i)})$.

The grid parameters θ specify the position of the grid and the width and height of the cells. We use maximum likelihood estimation to find the best set of parameters for each grid using the log likelihood function:

$$\ell(\theta) = \sum_{i=1}^{M} \log p(x^{(i)};\theta) = \sum_{i=1}^{M} \sum_{z^{(i)}} \log p(x^{(i)}, z^{(i)};\theta) \quad (3)$$

Since the number of grid rows and columns cannot be updated by maximizing a differentiable function, we learn these additional parameters by adding a row and/or column to the grid and comparing the likelihood values.

Finally, note that the algorithm in Table I learns the grid parameters such that the grid cell centers correspond to button detections centers. However, since we are interested in cropping out the button and the corresponding label, which is always located to the left of the button by the ADA guidelines, we shift the learned grid to the left such that the button detection cluster occupies the right half of the grid cell instead of its center, with the assumption that the label must then occupy the left half.

The overall algorithm thus allows us to learn a grid pattern with up to five grids with the appropriate number of rows and columns to correctly match each individual panel.

C. Optical Character Recognition

The first two steps of the pipeline, button detection and grid fit, produce estimates of the location of buttons and labels. Given these estimates, our next goal is to appropriately classify each button, e.g. `2nd floor`, `ground floor`, `alarm`, and so on. Consider the distribution of labels in Figure 6. Even though it is easy for a human to correctly identify most, if not all, of the labels, it becomes surprisingly difficult for a robot to do so autonomously using standard OCR algorithms.

For this part of the pipeline we use the open-source LeNet-5 convolutional neural network of Lecun et al. ([14]). However, this network was initially designed for and trained on handwritten digits, which were written in black ink on white background. Since this is quite dissimilar to our scenario, as evidenced by Figure 6, we had to (1) binarize

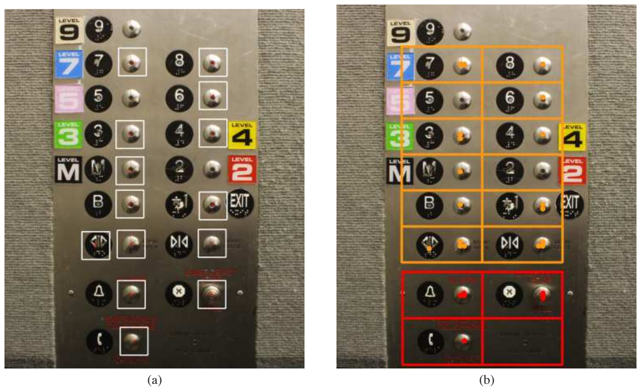


Fig. 5. EM grid fit step. (a) Button detection clusters. (b) Learned grid location using EM. Note the missing buttons in the grid 2 and 5 which are recovered but also note the false positive introduced in the bottom right corner of the grid.

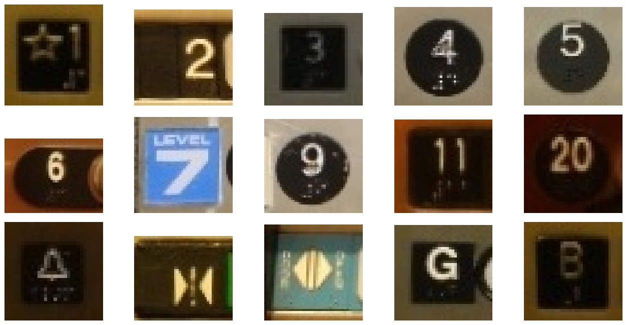


Fig. 6. Elevator button labels from our training and test images. Designing a character recognition system for these labels is a challenging task which we address in section III-C.

our data, i.e., process the images of elevator labels to obtain black images against white background to aid the OCR, and (2) retrain the neural network using sample binarized elevator labels instead of handwritten digits.

1) Image binarization: We want to extract from the label a binarized image of the segment, which corresponds to the alphanumeric character assigned to the button, to input into the OCR algorithm. A binarization of a given segment is an image where the pixel value is 1 if it belongs to the segment and 0 otherwise. Various binarization techniques have been proposed in the literature ([15], [16], [17]), many specifically tailored for OCR ([18], [19]). However, in our

TABLE I

THE EM ALGORITHM FOR FITTING A GRID TO A SET OF ELEVATOR BUTTON DETECTIONS

```
Parameters: $\theta$ = $\{o_x, o_y, \Delta_x, \Delta_y\}$ where
            $o_x$ = x-coordinate of grid
            $o_y$ = y-coordinate of grid
                 (center of top-left cell)
            $\Delta_x$ = width of grid cell
            $\Delta_y$ = height of grid cell

Given: (1) button detections $\{x^{(1)}, \ldots x^{(m)}\}$ where
           $x^{(i)} \in \mathcal{R}^4$ specifies the center, width
           and height of the detection
       (2) $n_r$, $n_c$ number of rows, columns in grid

Hidden: assignments $\{z^{(1)}, \ldots z^{(m)}\}$ of each detection
        to one of the grid cells
           $z^{(i)} \in \{1, 2, \ldots n_c n_r,$ outlier$\}$
.............................................................
Repeat until convergence {
    For each grid cell $j$:
        let $\mu_j$ be the coordinates of its center
        based on the grid parameters $\theta$
    E-step:
        For each detection i:
            $Q_i(z^{(i)}) = p(z^{(i)}|x^{(i)};\theta)$
                $\propto \begin{cases} \mathcal{N}(\mu_{z^{(i)}}|x^{(i)},\Sigma) \times \frac{(1-\epsilon)}{n_r n_c} & \text{if } z^{(i)} \in \{1,2,\ldots n_r n_c\} \\ \frac{\epsilon}{w_{image} h_{image}} & \text{if } z^{(i)} = \text{outlier} \end{cases}$
    M-step:
        $\theta := \text{argmax}\left(\sum_{i=1}^{m} \sum_{z^{(i)}} \mathcal{Q}_i(z^{(i)}) \times \log \frac{p(x^{(i)}, z^{(i)};\theta)}{\mathcal{Q}_i(z^{(i)})}\right)$
}
```

case we are considering a task where only 1-2 characters appear per image which means that no algorithms can be applied to learn the font of the text or the geometric position of the characters relative to each other.

From Figure 6 it becomes clear that simple thresholding algorithms will not work for binarizing these labels. Further, edge detection and similar techniques do not yield adequate results in practice, since reliably post-processing their output to obtain coherent components corresponding to individual characters is extremely challenging.

Instead, we use an unsupervised superpixel segmentation algorithm due to Felzenszwalb et al. [20] with four different parameter settings (c = 50, 100, 200, 500) to obtain four different segmentations of the image. We then input *all* of the segments obtained from each of the four segmentations into the OCR algorithm. The segmentation parameters were determined from the training set, and in Figure 7 we illustrate our approach and show the benefit of using the output from all four segmentations, instead of just choosing a single segmentation parameter.

After some basic post-processing to get rid of segments that are too large, too small, or circular (usually corresponding to button boundaries), we turn each segment into a binary image of the same size as the detection window.

2) Training the neural network: We trained our network to recognize the common labels found on elevator panels (`1`, `2`, `3`, `4`, `5`, `6`, `7`, `8`, `9`, `G`, `B`, `L`, `open`, `close`, `alarm`).[4] The training data is comprised of binary images of segments taken from the training set. Because some labels such as `9`s, `8`s, and `7`s did not appear enough in the training set, synthesized segments were generated by performing minor transformations on the available segments, such as eroding and dilating random parts of each segment. The network was also trained to recognize segments that were not a label (`NAL`), since many of the segments from the binarization will not correspond to actual alphanumeric labels.

For each segment classified, the network outputs a confidence value associated with the label. Because multiple proposed binarized images are created for each input button, OCR produces multiple character predictions per button. We then filter the output by throwing away segments that have low confidence values or that get classified as `NAL`, and use the relative locations of the remaining segments to decide on the most likely label. For example, if we detect multiple 1s and 2s in the image we want to use their relative locations and confidence values to distinguish between labels of `1`, `2`, `11`, `12`, `21` or `22`.

D. Consistency Enforcement

From the OCR algorithm, we have a series of imperfect observations for the button label states. For example, it's not uncommon to see `15`, `16`, `11`, `18` for four consecutive floor buttons. A person who needs to get to the `17th` floor immediately predicts that the corresponding button would be between `16` and `18` even if the label is missing entirely. We incorporated this insight into our algorithm using a Hidden Markov Model (HMM) [13].

From the button detection, grid fit and OCR steps, for each panel we obtain one or more disjoint grids corresponding to proposed button positions, and a label for each cell within these grids. We run an HMM to probabilistically enforce consistency between the observed labels.

The states in our HMM correspond to floors. The grid fit step attempts to fit rectangular grids around the buttons, but the buttons are not usually arranged in fully populated rectangular arrays (for example, there is often one or more cells in the upper right or bottom right of the grid that do not contain actual buttons). Thus we need to explicitly introduce an extra `blank` state that signifies the absence of a button in a grid cell, instead of forcing each grid cell to necessarily be assigned to a floor.

The observations in our HMM are the character strings returned by OCR. The *emission probability distribution* defines the probability that a label c is emitted from state s, or, in other words, the probability that, given an image of a button that leads to floor s (e.g. floor 15), OCR returns the label c

[4]In our training set, we do not include `0` (since it is too easily mistaken with the circular outline around many of the labels), the `stop` label (since it is often associated with a knob or keyhole and not a button), or `phone` label (since it does not appear often and is much like a `1` in appearance).

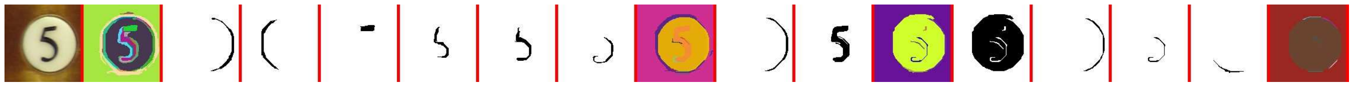


Fig. 7. Binarization step (best viewed in color). The first image is the original label. The other four non-binary images represent the output of the segmentation algorithm with four different parameter settings, with different colors corresponding to different segments. Each of these four segmentations is followed by binary images extracted from it using simple post-processing (discarding of segments that are deemed too large, too small, or circular). In this case only the second parameter setting produced a near-perfect segmentation of the character 5; however, all settings are needed for robust performance on varied elevator panels. Each of the obtained binarized images is then passed through OCR to extract a label.

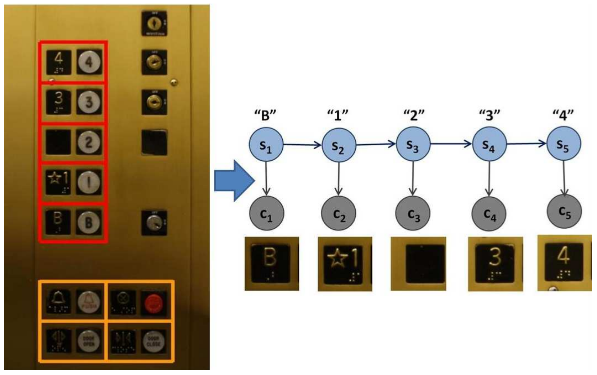


Fig. 8. An example of the grid fit and the HMM model (best viewed in color). Note that the observation from the OCR for the 2 will be NAL since the label, to the left of the button, is missing. The HMM will choose the most likely sequence to be one in which the NAL is changed to a 2.

(e.g."13"). The *transition probability distribution* defines the probability that a button for floor s_i is followed in the grid by a button for floor s_{i+1} (e.g. given a button for floor 15, the probability that the next button we encounter in the grid is for floor 21). These emission and transition conditional probability distributions are empirically estimated from the training data with Laplace smoothing. Figure 8 shows an graphical example of the HMM given the output from the grid fit and OCR for a given panel.

This learning procedure automatically captures interesting interdependencies between buttons such as a missing 13th floor, since the training data has many instances where the button for floor 14 often follows one for floor 12. It also learns that labels G, L or 1 all likely correspond to the same floor since they are often observed on the first floor button.

Further, during training this model observes the common errors made by other stages of the pipeline (e.g. that OCR tends to sometimes mistake 8s for 9s or that spurious labels of 1s are common due to the typical shape of segments from the binarization step) and learns to implicitly compensate for these mistakes. For instance, the model learns that 1s are often observed incorrectly and thus do not provide strong evidence for any particular floor; however, observing something more reliable, such as a two-digit number, is significantly more informative. Thus the HMM model provides additional robustness to the overall system.

Finally, given a test panel containing a grid of buttons, along with the output of OCR for each button, we use the Viterbi algorithm to find the most likely sequence of underlying states (floors) corresponding to those observations. We refer the reader to [13] for a detailed explanation of this standard algorithm.

IV. PERCEPTION RESULTS

To train and test our algorithm, we use a diverse set of 150 images of 61 distinct elevator panels taken with a high resolution (7 Megapixel) digital camera. To evaluate the performance of our approach, we randomly split the data into 100 images for training and 50 for testing, making sure that pictures from the same elevator do not appear in both the training and the test set. We hand label all the buttons with their location in the image, their character label, and their grid assignment. We then separately evaluate each step in our pipeline.

A. Sliding window button detection and clustering

To train the button detector, we use 998 positive examples (cropped buttons) and 20,000 negative examples cropped from the training set images. Since the button model often identifies keyholes and labels as buttons, we include images of keyholes and labels in the negative training set. We train an initial classifier, augment the set of negative examples with false positive detections obtained by running the sliding window detector on the training elevator panels, and retrain the button classifier.

The positive examples consist of three main types of buttons: circular plastic buttons, square plastic buttons, and circular metalic buttons. These types are visually very distinct from each other, and this variety allows the trained classifier to detect buttons on a variety of elevator panels. Figure 10(a) shows the PR curve for this model on the test panels. As is standard in computer vision [21], a detection is considered correct if its intersection with a groundtruth button divided by the union of their areas is greater than 50%, and at most a single detection per groundtruth button is considered correct.

Given the raw detections we first use adaptive thresholding to remove all low-probability detections. Intuitively, if a panel classification results in a large number of very high-probability detections we want to keep only the most confident of those; however, if a panel is such that the buttons are very difficult to detect, we want to make sure we do not discard all the detections, as we would with a fixed global probability thresholding method. Thus we determine the threshold by considering, for each panel, all detections of probability of at least 0.25, computing their standard deviation of the detection probability, and removing all detections with probability less than 3.0 standard deviations below the maximum probability.

For the clustering step, we use a confidence pixel map of the image created by summing the probabilities of all detections that lie over a pixel. We then compute the standard deviations of all values within this map and remove detections centered on pixels that have a probability less than one standard deviation from 0. As shown in Figure 10(a), the dynamic thresholding and clustering greatly improves the performance of the detector when compared to standard non-maximal suppresion followed by a fixed thresholding approach.

B. Grid fit

Next, we want to take advantage of the fact that buttons are typically arranged in a grid-like fashion to help recover any missed detections. As described in section III-B, we initialize grids based on the button detection clusters and then run the EM algorithm to more accurately fit these grids to our button detection data. We extract the proposed button locations from grids by choosing, for each cell, either the average center of all button detections present within it or simply the right half of the cell if no detections are present.[5]

The performance of each of these steps is reported in Table 10. Observe the drop in precision and increase in recall with the grid initialization and the EM steps. A drop in precision is expected because the EM grid fit algorithm assumes that buttons are arranged in fully populated rectangular grids, but often a row or column on the panel is not completely filled. Thus the overall number of proposed buttons increases. Despite introducing some false positives, the grid fit also recovers missed buttons in the grid (or false negatives) which were not found by the object detector, leading to an increased number of true positives and thus higher recall. It is more important to recover false negatives in the grid fit step, because false positives which are introduced will be eliminated if the OCR step does not classify the label as a valid alphanumeric character (Figure 5).

C. OCR and HMM

For each button we run the all the binarized segments from the label image through OCR and then automatically choose which character(s) to output for the button based on the returned OCR confidence values. Our dictionary includes `1-9`, `G`, `L`, `B`, `open`, `close`, and `alarm`. Our OCR dictionary includes only the alphabetic characters `G`, `L`, and `B` because there are not enough instances of other characters to build them into the model. Thus our algorithm will assign a `NAL` to the occasional instances of `P`, `DH`, etc.

To evaluate the performance of the image binarization and OCR, we use the ground truth labels for each panel. We use the resulting OCR classifications and the ground truth grid assignments to extract sequences of labels, to input into the HMM. Table 10(b) gives the recall for the OCR only and the OCR followed by HMM. It is clear that the HMM is able to correct some of the mislabeled buttons from the OCR step, resulting in improved performance.

[5]Recall that by construction each cell corresponds to a button and an adjacent label, and labels are assumed to be to the left of the buttons following ADA guidelines.

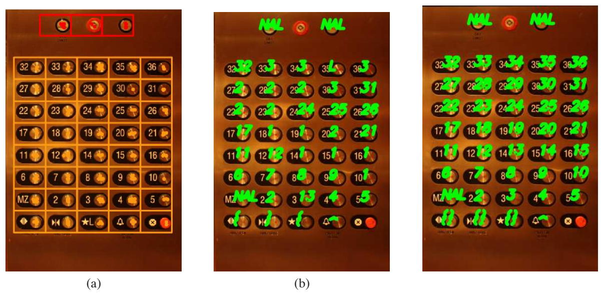


Fig. 9. (a) Learned grid location using EM. (a) OCR label. (c) HMM label correction. Note that OCR misclassifies over 20/35 of the floor buttons and HMM is able to correct for all but one.

Although the grid fit step introduces false positives due to elevator buttons not lying in fully populated grids, the adverse effect of these false positives is reduced by not including cells in the HMM input for which (1) OCR classified as `NAL` and (2) the button detector did not find a detection. Figure 9 shows an example of the OCR and HMM output for a given EM grid fit.

Figure 10(c) shows the results for all the steps along the pipeline. To evaluate the button detection and EM steps, we consider a classification as correct based on the button ground truth for the given panel. For the OCR and HMM steps, both the button and label classification must match that in the ground truth. The OCR performance is obviously lower than that shown in Figure 10(b), since the grid fit step does not produce perfectly cropped out labels like the ground truth, but HMM corrects some of mislabeled buttons. Overall, the entire pipeline, from the sliding window detector to the HMM, was able to perfectly detect, localize, *and* label 86.2% of the buttons in the images.[6]

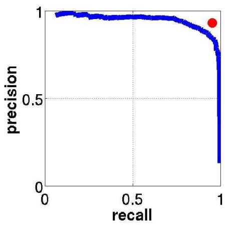
V. EXPERIMENTAL RESULTS

We demonstrate our perception algorithms on the STAIR (STanford AI Robot) mobile robot platform in a number of experiments where the robot was commanded to go to a given floor. The robot was required to autonomously locate the appropriate button and successfully press it in order for the trial to be considered a success.

A. Hardware

Our robotic platform consists of a Neuronics Katana450 5-DOF robotic arm, with angle gripper, mounted on a Segway base. The vision system consists of a Canon SX100-IS digital camera and a high resolution 3D sensor consisting of a Point Grey Research Flea2 camera and a rotating laser line scanner (for a detailed description of this sensor, see [6]). The depth data was used to provide only the 3D location of a button for manipulation but not recognition. The Canon camera was used to provide higher resolution images for the optical character recognition. The 3D sensor (Flea2 camera + laser) was calibrated with the Canon camera using a

[6]In computing the statistics listed for OCR and HMM, we consider buttons which are part of our OCR dictionary, as well as `10`, `20`, etc (even though we our dictionary does not include `0`). We do not include labels for buttons which were labeled as `NAL` in the ground truth.



(a)

Method	Accuracy
OCR only	0.836
OCR + HMM	0.884

(b)

Method	Recall	Precision	F-score
Post-processed detections	0.955	0.932	0.944
EM	0.963	0.841	0.898
OCR	0.742	0.709	0.725
HMM	0.862	0.881	0.871

(c)

Fig. 10. Results on the offline dataset of 50 test images. (a) The blue curve is the precision-recall curve for button detections from the sliding windows algorithm after running non-maximal suppression, which is standard in the literature. The performance using our adaptive thresholding method with clustering is 0.955 recall and 0.932 precision, and is shown in red. (b) Results of the OCR and HMM algorithm on labeling the ground truth detections (686 labels total). (c) Results for each step along the pipeline, starting with unlabeled images of elevator panels.

standard camera calibration procedure which estimates the transformation between the two sensors given a number of correspondences between scene and image points [22]. The vision-manipulator system was calibrated using a procedure describe in [23], resulting in an average error of 0.7 mm.

B. Perception

We tested our algorithms on the robotic platform in 14 trial runs in 3 elevators in 2 buildings which the robot had not seen before. The robot was placed in front of the panel at a distance within which the manipulator could reach the panel, and was commanded to press a randomly chosen floor button.

The path planning for the manipulator was simplified by assuming that the space between the robot and the wall was free of obstacles. Thus simple inverse kinematics could be used to convert the desired 3D positions for the end-effector to joint angle configurations. In computing the desired location for the end-effector, we applied the constraint that the end-effector should move in a linear fashion when pressing the button to ensure that the elevator is successfully activated.

C. Results

The perception algorithm correctly identified the location of the appropriate button in all 14 trials, and the robot succeeded in pressing the button in all but one of the trials. See *http://www.stanford.edu/∼ellenrk7/Elevators* for a video showing clips from these experiments.

VI. CONCLUSION

In this paper, we considered the challenge of enabling a mobile robot to autonomously operate unkown elevators. We focus on perception, which is the key component in tackling this problem: detecting, localizing, and correctly labeling the buttons on an interior button panel of previously unseen elevators. We validate our algorithm both on a large dataset of diverse elevator panel images as well as on a robot platform that was able to correctly analyze a previously unseen elevator panel in all of our trial runs, and correctly manipulate the desired button in all but one run.

REFERENCES

[1] B. Kuipers and A. Stroupe, "The AAAI-2002 robot challenge," *AI Magazine*, vol. 24, pp. 65–72, Spring 2003.

[2] R. Simmons *et al.* (2003) Grace: An autonomous robot for the AAAI robot challenge. [Online]. Available: http://www.cs.cmu.edu/∼reids/papers/grace.pdf

[3] J. Miura, K. Iwase, and Y. Shirai, "Interactive teaching of a mobile robot," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, Apr. 2005, pp. 3378–3383.

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

[5] Singapore Defense Science and Technology Agency (DSTA). (2008, Aug.) Techx challenge rules. [Online]. Available: http://www.dsta.gov.sg/images/stories/TechX/pdf/techx_challenge_rules_11aug08a.pdf

[6] M. Quigley, S. Batra, S. Gould, E. Klingbeil, Q. Le, A. Wellman, and A. Ng, "High accuracy 3d sensing for mobile manipulation: Improving object detection and door opening," in *IEEE International Conference on Robotics and Automation*, 2008, in press.

[7] A. Oliva and A. Torralba, "The role of context in object recognition," *Trends in Cogn Sci*, Nov. 2007.

[8] S. K. Divvala, D. Hoiem, J. H. Hays, A. A. Efros, and M. Hebert, "An empirical study of context in object detection," in *Computer Vision and Pattern Recognition (CVPR)*, 2009.

[9] G. Heitz and D. Koller, "Learning spatial context: Using stuff to find things," in *ECCV*, 2008.

[10] A. D. A. Regulations. [Online]. Available: http://www.ada.gov/reg3a.html#Anchor-11540

[11] A. Torralba, K. P. Murphy, and W. T. Freeman, "Sharing visual features for multiclass and multiview object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, 2007.

[12] S. Gould, O. Russakovsky, I. Goodfellow, P. Baumstarck, A. Y. Ng, and D. Koller. (2008) The STAIR vision library (v2.1). [Online]. Available: http://ai.stanford.edu/ sgould/svl

[13] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY: Springer Science+Business Media, 2006.

[14] Y. B. Y. LeCun, L. Bottou and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE, 86(11):*, vol. 86, no. 11, pp. 2278–2324, 1999.

[15] "A threshold selection method from gray-level histograms," *IEEE Trans. Systems, Man and Cybernetics*, vol. 17, no. 5, pp. 62–66, 1979.

[16] "Survey over image thresholding techniques and quantitative performance evaluation," *Journal of Electronic Imaging*, vol. 13, no. 1, pp. 146–165, 2004.

[17] "An efficient iterative algorithm for image thresholding," *Pattern Recognition Letters*, vol. 29, no. 9, pp. 1311 – 1316, 2008.

[18] "OCR binarization and image pre-processing for searching historical documents," *Pattern Recognition*, vol. 40, no. 2, pp. 389 – 397, 2007.

[19] Y. Liu and S. N. Srihari, "Document image binarization based on texture features," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 5, pp. 540–544, 1997.

[20] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *IJCV*, vol. 59(2), 2004.

[21] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. (2007-2010) The PASCAL Visual Object Classes Challenge Results.

[22] A. V. Emanuele Trucco, *Introductory Techniques for 3-D computer vision*. Upper Saddle River, NJ: Prentice Hall, 1998.

[23] Q. Le, "Joint calibration of multiple sensors," in *Proceedings of the 2009 IEEE International Conference on Intelligent Robots and Systems*, Oct. 2009, pp. 3651–3658.